Notas sobre
Dinámica de Fluidos Computacional

Rev. 0.5.5

José Ignacio Cardesa Dueñas

ji.cardesa@.upm.es

9 de septiembre de 2014

Este documento está publicado según la siguiente licencia:

GNU Free Documentation License

Copyright © 2014 José Ignacio Cardesa Dueñas. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Estos apuntes están basados sobre los apuntes de Adrián Lozano Durán, que se pueden descargar aquí: http://torroja.dmt.upm.es/adrian/, de los cuales han sido editadas y quitadas ciertas partes para adaptarlos al contenido del curso de grado de la ETSI Aeronáuticos, año académico 2014-2015, asignatura MEF-DFC, Universidad Politécnica de Madrid.

Índice

Índice **1**

1 Computación Científica **3**
1.1 El ordenador como herramienta para resolver problemas matemáticos 3
1.2 Representación de números 5
1.2.1 Representación y aritmética de punto flotante 6
1.2.2 Round off error o error de redondeo 7
1.3 Introducción a los lenguajes de programación 9
1.4 Arquitectura del ordenador 10
1.4.1 Procesador 11
1.4.2 Memoria 11
1.4.3 Redes 14
1.5 Introducción al cálculo en paralelo 14
1.5.1 ¿Cuándo es necesario? 15
1.5.2 Paradigmas de programación en paralelo 15

2 Planteamiento del problema CFD **19**
2.1 Ideas generales de la discretización temporal 20
2.2 Ideas generales de la discretización espacial 20
2.2.1 Clasificación de métodos de discretización espacial . . . 21
2.2.2 Clasificación de mallas 22
2.2.3 Generación de mallas 24

3 Discretización temporal **27**
3.1 Problema de condiciones iniciales 27
3.2 Clasificación de esquemas numéricos 28
3.3 Obtención de esquemas numéricos 30
3.4 Errores de la solución numérica 37
3.5 Análisis de esquemas numéricos 40
3.5.1 Existencia y unicidad de la solución de la ecuación diferencial 40

3.5.2 Estabilidad de la solución de la ecuación diferencial . . 41
3.5.3 Consistencia, estabilidad y convergencia del esquema numérico 44

4 Estrategias de resolución de las ecuaciones de la Mecánica de Fluidos 50
4.1 Formulación con presión 50
4.1.1 Método de proyección 51
4.2 Formulación sin presión 53
4.2.1 Vorticidad-función de corriente 54
4.2.2 Vorticidad-velocidad 56

5 Discretización espacial: diferencias finitas 57
5.1 Obtención de esquemas 59
5.2 Análisis de errores 61
5.2.1 Error de truncación 61
5.2.2 Error de disipación y dispersión 62
5.2.3 Análisis de onda modificada 63
5.3 Estabilidad de la discretización espacio-temporal 65
5.4 Mallas de colocación y mallas staggered 68
5.5 Aplicación a problemas 1D 70
5.5.1 Ecuación de onda 71
5.5.2 Ecuación del calor 74
5.5.3 Ecuación de Burgers viscosa 78
5.6 Aplicación a problemas 2D: la cavidad 84
5.6.1 Planteamiento del problema 84
5.6.2 Discretización temporal 85
5.6.3 Discretización espacial 85
5.6.4 Organización matricial de las ecuaciones 87
5.6.5 Condiciones de contorno 88
5.6.6 Código 89

Capítulo 1

Computación Científica

1.1 El ordenador como herramienta para resolver problemas matemáticos

El ordenador es una máquina extremadamente potente pero también inútil si no se le proporcionan las instrucciones adecuadas. Es importante dejar a un lado la idea de que *ésto es fácil porque el ordenador lo resuelve*. El ordenador es tonto, sólo se limita a ejecutar las órdenes que le damos, ni más ni menos. Para él es indiferente darnos una solución donde un fluido se mueve con velocidades del orden de metros por segundo o por el contrario varias veces la velocidad de la luz. Por eso, es fundamental el juicio crítico de los datos procedentes de un ordenador tanto en CFD como en cualquier otra disciplina. Por otro lado, hay que tener en cuenta que calcular la solución del problema no es resolver el problema, sino solo un primer paso para entender el porqué de dicha solución.
Ciencia Computacional o Computación Científica (Computational Science, no confundir con Computer Science) es la disciplina encargada de construir y analizar las herramientas necesarias para resolver problemas matemáticos mediante el uso de ordenadores. La principal limitación impuesta por el ordenador es que es una máquina finita y discreta con la cual deseamos resolver problemas que muchas veces son continuos. De forma muy general, podemos clasificar la resolución de problemas en:

- **Resolución simbólica o álgebra computacional.**
  Consiste en el cálculo exacto de expresiones que contienen variables a las cuales no se le ha atribuido ningún valor numérico y son manipuladas de forma simbólica para dar lugar a soluciones exactas. Los cálculos se realizan con precisión arbitraria (sin errores de truncación

ni redondeo) y utilizando símbolos o variables. En muchos campos de investigación es necesario procesar largas expresiones algebraicas lo que resulta un trabajo largo y tedioso. Por ello, siempre que sean perfectamente conocidos los pasos que hay que seguir para obtener el resultado, se puede aplicar la resolución simbólica por ordenador. Aún así, no está exento de problemas por la inevitable existencia de bugs (errores) en los códigos y la dificultad de obtener resultados lo suficientemente simplificados. Los inicios del software del álgebra computacional comienza en 1964 con ALPAK, desarrollado por Bell Labs y seguido de FORMAC de IBM. Actualmente algunos de los software más comunes son Maple y Mathematica entre otros.

- **Resolución numérica. Cálculo numérico.**
  Se trata de la concepción y estudio de métodos de cálculo que aproximen la solución de problemas previamente formulados matemáticamente mediante el uso de algoritmos. Definimos algoritmo como secuencias finitas de operaciones algebraicas y lógicas que producen una solución al problema dado. En este caso el resultado final no es simbólico sino valores numéricos. Existen multitud de problemas que pueden ser resueltos mediante el cálculo numérico tales como integración definida, derivación, interpolación, sistemas de ecuaciones algebraicas, ecuaciones diferenciales ordinarias, ecuaciones diferenciales en derivadas parciales (CFD). Las soluciones son aproximadas pero se pueden resolver aquellos problemas que no tienen solución analítica o que en el caso de tenerla es difícil de obtener. El CFD se puede entender como aquel conjunto de herramientas del cálculo numérico aplicadas a la resolución de problemas fluido dinámicos.

La siglas CFD son el acrónimo de Dinámica de Fluidos Computacional (Computational Fluid Mechanics). La física de los fluidos puede ser expresada en términos de ecuaciones diferenciales ordinarias o integro-diferenciales difíciles de resolver analíticamente excepto en casos muy concretos de poco interés práctico. Para obtener la solución aproximada numéricamente es necesario discretizar las ecuaciones diferenciales en ecuaciones algebraicas que serán resueltas mediante los algoritmos apropiados ejecutados por lo general en ordenadores. Entre las grandes ventajas que ofrece el CFD se encuentra el bajo coste que presentan la simulación de prototipos en comparación con ensayos de modelos a escala real o reducida. Además existe la libertad para imponer condiciones de contorno y obtenemos la información de todas las variables en gran cantidad de puntos del espacio, algo imposible en experimentos. Hay que tener en cuenta que muchas veces es complicado fijar los

$2/6 \to 0.333333$	$2/6 \to 1/3$
$2+3 \to 5$	$x+2x \to 3x$
$\cos(3.14159) \to -0.999999$	$\cos(\pi) \to -1$
	$\sin(2x) \to 2\sin x \cos x$
	$\frac{d\,x^2}{d\,x} \to 2x$
$\int_0^{1/2} \frac{x}{x^2-1} d\,x \to 0.1438$	$\int \frac{x}{x^2-1} \to \frac{\ln\lvert x^2-1\rvert}{2}$
	$a^2-b^2 \to (a+b)(a-b)$

Figura 1.1: Tabla con ejemplos de cálculos realizados mediante cálculo numérico (columna de la izquierda) o simbólico (columna de la derecha).

parámetros adimensionales en los experimentos para que coincida con los del caso que se quiere analizar, especialmente cuando hay que imponer varios de ellos como por ejemplo el número de Reynolds y número de Froude. Por otro lado, el CFD también presenta limitaciones. Uno de los inconvenientes más importantes es lo costoso que resulta resolver todas las escalas de las ecuaciones de Navier-Stokes cuando el fluido se encuentra en régimen turbulento, lo que obligar a reducir el tamaño de la simulación usando modelos en las ecuaciones que pueden dar lugar a soluciones no solo cuantitativamente incorrectas sino también cualitativamente.

1.2 Representación de números

Los computadores manejan datos representados como una secuencia discreta de bits. Cada bit puede estar en dos valores diferentes a los que simbólicamente se asocian los estados 0 y 1, por ello, utilizan de forma natural el sistema en base 2. Los datos almacenados pueden ser numéricos y no numéricos. Los números se pueden representar en el sistema de numeración binario y ésta es la base para la representación de números en los ordenadores. Puesto que cualquier entero dado sólo tiene un número finito de dígitos, se pueden representar exactamente todos los números enteros por debajo de un cierto límite. Los números reales no son numerables y son más complicados dado que se necesita una cantidad infinita de dígitos para representar la mayoría de ellos, sin importar qué sistema de numeración utilicemos. En general, con n bits podemos representar 2^n números. Los números enteros se suelen almacenar como *punto/coma fijo* mientras que los reales se guardan con *punto/coma flotante*.

IEEE simple precision

Signo	Exponente	Mantisa
1 bit	8 bits	23 bits

IEEE doble precision

Signo	Exponente	Mantisa
1 bit	11 bits	52 bits

Figura 1.2: Esquema de los bits asignados al signo, mantisa y exponente en los formatos de precisión simple y doble según el estándar IEEE 754.

1.2.1 Representación y aritmética de punto flotante

Cuando disponemos de n bits, tenemos que decidir qué conjunto finito de números vamos a representar. En la aritmética de punto flotante los números se representan repartiendo los n bits entre una mantisa (el significando), un exponente y un bit para el signo, que no es más que una forma de notación científica. De esta manera conseguimos representar un gran rango de números reales con un número finito de bits.
El estándar que define cómo se asignan los bits a la mantisa, signo y exponente y la forma de operar con ellos es el IEEE 754[1]. El formato IEEE 754 establece la normalización de la mantisa (el número antes del punto no se suele almacenar) y define la precisión simple con el uso de 32 bits y la doble con 64 bits. Además establece los tamaños de la mantisa y exponente y los criterios de redondeo (redondeo al más próximo con desempate al par). Algunas combinaciones se reservan para representaciones especiales como Inf (infinito positivo), -Inf, (infinito negativo) ó NaN (Not a Number). Definimos la *precisión del sistema* en punto flotante como el número t de bits de la mantisa que está íntimamente ligado al número de *cifras significativas*. Una mantisa de t cifras en binario cumple

$$2^{-t} \approx 10^{-m} \tag{1.1}$$

donde m son las cifras significativas en sistema decimal. Por ejemplo, en simple precisión para $t = 23$ tenemos $2^{-23} \approx 10^{-7}$ que implica 7 cifras significativas y en doble precisión con $t = 52$ tenemos $2^{-52} \approx 10^{-16}$ que da lugar a

[1]IEEE es una abreviación de *Institute of Electrical and Electronic Engineers*, una sociedad profesional de ingenieros y científicos de Estados Unidos. El estándar para la aritmética en punto flotante está recogido en la referencia 754.

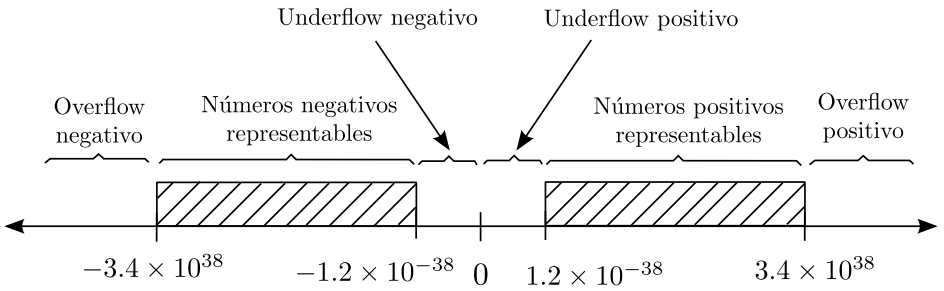


Figura 1.3: Representación de números en punto flotante para simple precisión en el estándar IEEE 754.

16. Otro concepto importante es la *precisión de la máquina* o ϵ *de la máquina* definido como el menor número que cumple

$$\epsilon + 1 \neq 1. \tag{1.2}$$

Representa la exactitud relativa de la aritmética en punto flotante y es consecuencia del redondeo. Decimos que ocurre *underflow* cuando el resultado de una operación es menor en magnitud que el número más pequeño que puede ser almacenado por el ordenador. Normalmente el resultado se redondea a cero. Por el contrario, decimos que ocurre *overflow* cuando el resultado de una operación es mayor en magnitud al mayor número que puede representar el ordenador. Normalmente se redondea el resultado a $\pm Inf$. Nótese que en la representación de punto flotante el espaciado entre números es mayor cuanto mayor es la magnitud del número. El ϵ *de la máquina* puede ser entendido como un underflow en la mantisa, mientras que el underflow y overflow están relacionados con el exponente.

1.2.2 Round off error o error de redondeo

La representación en el ordenador de números no enteros en punto flotante se hace con un número fijo de bits. Ésto significa que la mayoría de los números no enteros no se pueden representar sin cometer un error que normalmente se conoce como *roundoff error* o error de redondeo. Existe, por lo tanto, un error simplemente por el hecho de almacenar un número. Además, la mayoría de los cálculos (sumas, restas, multiplicaciones, divisiones...) con números en punto flotante producirán más errores de redondeo. En la mayoría de las situaciones estos errores serán pequeños, pero en una larga cadena de cálculos hay un alto riesgo de que los errores se acumulen y contaminen gravemente el resultado final. Es importante ser capaz de reconocer cuándo un cálculo dado va a ser propenso a este tipo de problemas y saber si el resultado es

fiable. Consideremos un número a y una aproximación $\tilde{a}$. Vamos a definir dos formas de medir el error de dicha aproximación.

- Error absoluto: $|a-\tilde{a}|$. Es la forma más obvia de medir el error. Presenta ciertos inconvenientes, por ejemplo, para $a = 100$ y $\tilde{a} = 100.1$ el error absoluto es el mismo que para $a = 1$ y $\tilde{a} = 1.1$, cuando parece intuitivo pensar que el error cometido es mayor en el último caso. Por ello, en ciertas ocasiones es mejor utilizar el error relativo.

- Error relativo: $|a-\tilde{a}|/|a|$, que supone escalar el error absoluto obtenido con el tamaño del número que es aproximado. En el ejemplo anterior los errores relativos serían, 10^{-3} y 0.1 lo cual resulta más razonable. Un propiedad importante del error relativo es que cuando

$$r = \frac{|a - \tilde{a}|}{|a|} \approx 10^{-m}, \tag{1.3}$$

  con m un entero, entonces el número de cifras que tienen en común a y $\tilde{a}$ es aproximadamente m y por lo tanto la precisión del sistema nos indica indirectamente el error relativo que se comete al almacenar un número en punto flotante.

Los errores en la aritmética de punto flotante son mucho más sutiles que los errores en aritmética de enteros. A diferencia de los números enteros, los números de punto flotante pueden estar ligeramente mal. Un resultado que parece ser razonable contiene errores y puede ser difícil juzgar cuán grandes son. Tal y como se mencionó en la sección anterior, en la mayoría de los ordenadores los números se representan en punto flotante y la aritmética se realiza de acuerdo con la norma IEEE 754, cuidadosamente diseñada para proporcionar un buen control de errores de redondeo. Sin embargo, el uso de números en punto flotante conduce inevitablemente a errores en la mayoría de los casos de interés práctico. En general, las operaciones de adición y sustracción producen mayores errores que el producto y la división.
El esquema general del proceso de adición (o sustracción) es:

- Partimos de dos números reales a y b con $|a| > |b|$ y queremos realizar la operación $c = a + b$

- Escribimos a en forma normalizada $a = \alpha \times 10^n$ y b de tal manera que tenga el mismo exponente $b = \beta \times 10^n$.

- sumamos los significantes $\gamma = \alpha + \beta$.

- El resultado $c = \gamma \times 10^n$ es redondeado y normalizado.

El estándar exige que el resultado de las operaciones sea el mismo que se obtendría si se realizasen con precisión absoluta y después se redondease. Por ello, es el último paso (redondeo) el que puede dar lugar a grandes errores cuando se suman dos números de tamaños muy diferentes dado que la mantisa que se utiliza para guardar el resultado final es finita. El problema es similar cuando se sustraen dos números muy cercanos. En general si tenemos una mantisa con m cifras significativas, $a + b = a$ cuando b es más de m órdenes de magnitud menor que a, es decir, no es posible percibir el cambio de a al añadir b. En el caso de la sustracción tendremos problemas cuando los dos números sean muy próximos ya que la mayor parte de la cifras de la mantisa se cancelan. Aunque la operaciones de multiplicación y división parezcan más complicadas, los errores cometidos son menores. Al multiplicar dos número el proceso se reduce a multiplicar sus significantes y sumar los exponentes. Tras ello, se normaliza el resultado. La multiplicación y división de números en punto flotante no conduce a la pérdida de cifras significativas siempre y cuando los números se encuentren en el rango del modelo de punto flotante utilizado. En el peor de los casos el último dígito del resultado puede ser incorrecto.

1.3 Introducción a los lenguajes de programación

Un lenguaje de programación es un lenguaje artificial diseñado para comunicar instrucciones (algoritmo) a una máquina, generalmente un ordenador. A grandes rasgos podemos clasificar los lenguajes de programación en:

- Máquina: código binario, directamente entendible por el ordenador.
- Bajo nivel: instrucciones en códigos alfabéticos, intrínsecamente relacionado con el lenguaje máquina (ensamblador).
- Alto nivel: sentencias con palabras similares al lenguaje humano . Es el que se suele utilizar para programar las herramientas de CFD y en general todo tipo de software y que a su vez pueden ser:
  - Estáticos: C, FORTRAN...
  - Dinámicos: Octave, Matlab, Python...

El desarrollo de los Lenguajes de programación ha sido impresionante en los últimos 60 años. Los primeros lenguajes de alto nivel aparecieron en la década de los 50 con FORTRAN (Formula Translating System, creado por

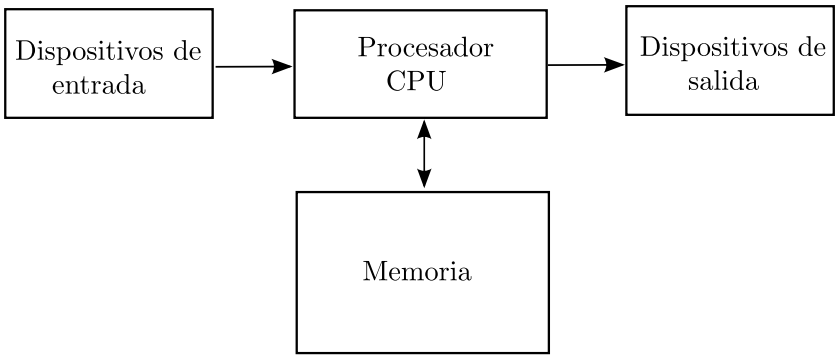


Figura 1.4: Arquitectura de von Neumann. Es el modelo que siguen a grandes rasgos casi todos los ordenadores actuales.

John Backus), COBOL, LISP... Después surgirían otros como Algol, Basic, C, Pascal, C++... Para dar lugar a los más actuales y modernos como C#, Python, Java, PHP... Algunos de los lenguajes de programación más usados actualmente en el cálculo numérico son: FORTRAN, C, (estáticos), Octave, Matlab, Python (dinámicos). En otras ocasiones se utilizan programas ya compilados como OpenFoam.
Muchas veces, en el diseño de un algoritmo se utilizan diagramas de flujo y pseudocódigos como lenguaje intermedio entre el lenguaje de programación y el lenguaje natural.

1.4 Arquitectura del ordenador

La arquitectura del ordenador es un tema amplio y complicado en el que evidentemente no deseamos entrar en gran detalle. Sin embargo, los códigos CFD que usamos acaban ejecutándose en un ordenador y es necesario tener un idea general de su funcionamiento. A continuación resaltamos los aspectos más importantes relacionados con el uso de programas CFDs.
Casi todos los ordenadores siguen a grandes rasgos el esquema propuesto en el modelo de von Neumann. Los ordenadores con esta arquitectura constan de cinco partes: La unidad aritmético-lógica (ALU) que junto con la unidad de control forman el procesador, la memoria, un dispositivo de entrada/salida y el bus de datos que proporciona un medio de transporte de los datos entre las distintas partes.

Intel	I7 3930K	5Ghz	104 GFLOPS
AMD	Phenom II 1090t	4.2Ghz	80 GFLOPS
Intel	Core i5-2320	3.0Ghz	44 GFLOPS
Intel	Core 2 Duo E6550	2.3Ghz	6 GFLOPS
Intel	Atom N455	1.66 GHz	1 GFLOPS

Cuadro 1.1: FLOPS para diferentes procesadores.

1.4.1 Procesador

El procesador o CPU es el encargado de ejecutar los programas. Sólo ejecuta instrucciones programadas en lenguaje de máquina, realizando operaciones aritméticas y lógicas simples, tales como sumas, restas, multiplicaciones, divisiones, lógicas binarias y accesos a memoria.
Un parámetro importante del procesador son los FLOPS (FLoating-point Operations Per Second) que indica el número de operaciones en punto flotante que el procesador es capaz de realizar por segundo. Los ordenadores de sobremesa actuales tienen del orden de Giga FLOPS. La tabla 1.1 recoge algunos procesadores y una estimación sus respectivos FLOPS.
En la práctica, se puede estimar cuál será la capacidad de cálculo de los procesadores dentro de unos años usando la Ley de Moore: *el número de transistores en un procesador (íntimamente ligado a la capacidad de cálculo) se duplica aproximadamente cada 2 años*. Se trata de una observación, una ley empírica formulada por Gordon E. Moore, en 1965, cuyo cumplimiento se ha mantenido hasta nuestros días.
Un procesador con muchos FLOPS no es la solución a todo problema y en general un buen algoritmo reduce en mayor medida el tiempo de cálculo que disponer de procesadores muy rápidos. Además, en los últimos años el sector informático está dando mucha importancia a factores como el consumo de electricidad y el rendimiento por vatio. Los procesadores de ordenadores de sobremesa suelen consumir entre 60 y 100 Watios, mientras que los de los portátiles consumen entre 20 y 40 Watios. Hay que tener en cuenta que en el cálculo en paralelo (ver siguiente apartado) se pueden llegar a usar cientos de miles de procesadores a la vez y el consumo se convierte en un factor importante.

1.4.2 Memoria

El correcto uso de la memoria es un tema fundamental para obtener buenos rendimientos de los códigos CFD. La figura 1.6 muestra las diferentes jerar-

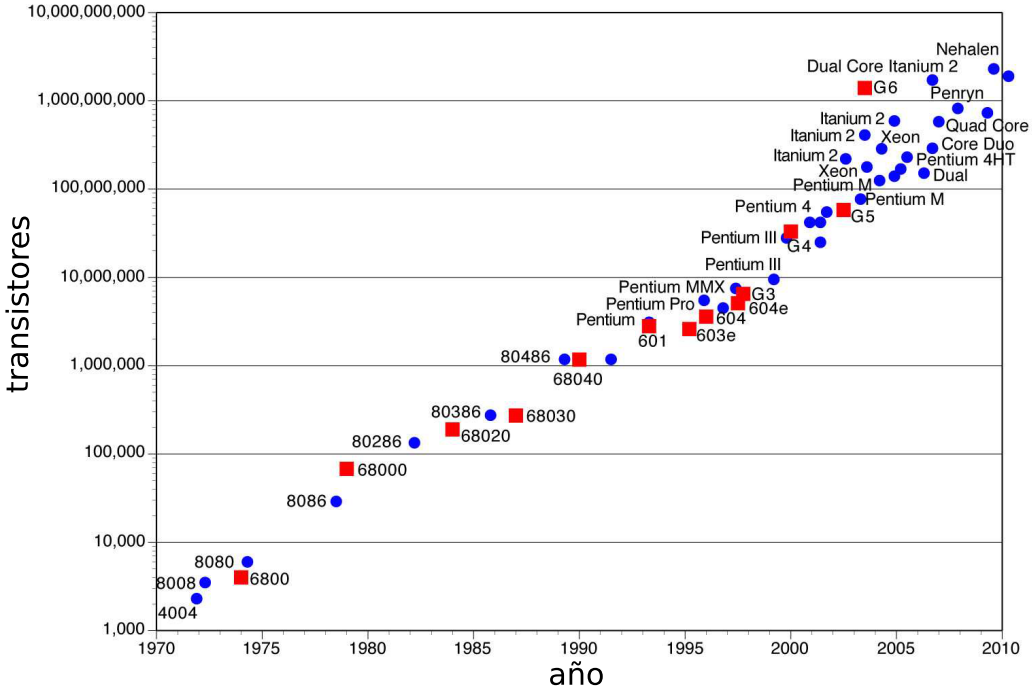


Figura 1.5: Ley de Moore. El número de transistores en un procesador se duplica aproximadamente cada dos años.

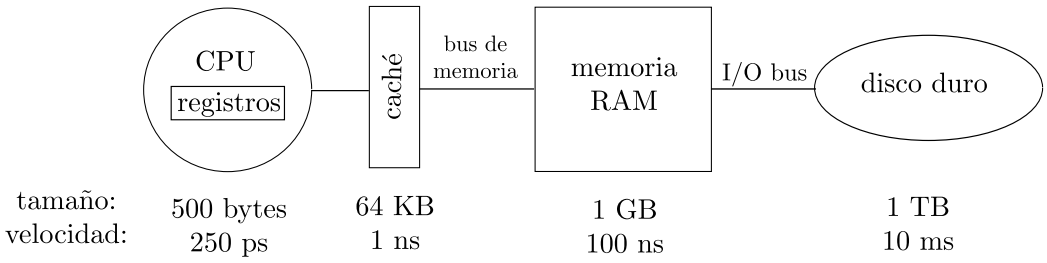


Figura 1.6: Jerarquía de memorias en un ordenador. Los tamaños y velocidades dados son valores de referencia.

quías de memorias en un ordenador: Disco duro, RAM y caché.

- Memoria caché:
  Es la memoria más rápida de la cual dispone el procesador. Se utiliza para tener alcance directo a datos que predeciblemente serán utilizados en las siguientes operaciones, sin tener que acudir a la memoria RAM, reduciendo así el tiempo de espera para adquisición de datos. Casi todos los procesadores poseen la llamada caché interna de primer nivel o L1 encapsulada en el procesador. Los más modernos incluyen también en su interior otro nivel de caché, más grande, aunque algo menos rápida, es la caché de segundo nivel o L2 e incluso los hay con memoria caché de nivel 3, o L3.

- Memoria RAM:
  Es la memoria de acceso aleatorio. Es una memoria rápida que permite acceder a los datos en cualquier orden. En ella se almacenan todos los programas que se están ejecutando. Tanto la memoria RAM como la caché son volátiles, y pierden la información si se dejan de alimentar/energizar.

- Disco duro:
  Sistema de almacenamiento digital no volátil. Suele ser la memoria más lenta de todas, pero la que tiene mayor tamaño.

Es importante resaltar que cuanto más lejos nos movemos del procesador, el nivel de memoria se convierte en 10 veces más lento (de picosegundos a milisegundos) y 1000 veces más grande (de bytes a terabytes).
Normalmente el programador puede controlar directamente el flujo entre la memoria RAM y el disco duro pero no entre la memoria RAM y la caché, aunque dicho control se puede hacer indirectamente siguiendo ciertas pautas de programación.

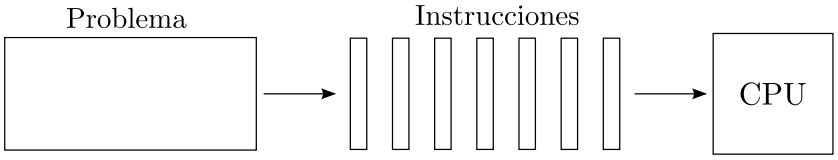


Figura 1.7: Esquema de ejecución de un programa en serie.

Existe una forma equivalente a la Ley de Moore para el almacenamiento en disco duro llamada Ley de Kryder: *la cantidad de bits por unidad de volumen en un disco duro se duplica aproximadamente cada 13 meses*. Se trata de una ley experimental enunciada por Mark Kryder (ingeniero de Seagate Technology). Una consecuencia de comparar la Ley de Moore con la Ley de Kryder es que la capacidad de almacenamiento crece más rápidamente que la de procesamiento. Además, los tiempos de acceso a memoria también se han reducido más lentamente lo que plantea problemas de cuello de botella en el flujo de datos entre el disco duro y el procesador.

1.4.3 Redes

En algunas ocasiones los códigos CFD no son ejecutados en un solo ordenador sino que es necesario el cálculo en paralelo mediante el uso de un array de ordenadores conectados en red. En esos casos es, la red pasa a ser, junto con el procesador y la memoria, otro elemento fundamental a tener en cuenta.

1.5 Introducción al cálculo en paralelo

Tradicionalmente, los programas se han desarrollado para el cálculo en serie, es decir, están preparados para ejecutarse en un ordenador con un único procesador. El problema es dividido en un conjunto de instrucciones que son ejecutadas secuencialmente.
El cálculo en paralelo consiste en usar múltiples recursos simultáneamente para resolver un problema dado. El problema es dividido en partes independientes que son ejecutadas simultáneamente en varios procesadores. Las figuras 1.7 y 1.8 muestran los esquemas de ejecución en serie y paralelo.
El cálculo en paralelo se realiza en los llamados *centros de supercomputación*. En ellos, arrays de nodos de cálculo se conectan entre sí mediante una red rápida. En la web *http://www.top500.org* se pueden encontrar estadísticas y datos interesantes sobre estos centros, como su uso por paises, las aplicaciones, sistemas operativos que usan... La figura 1.9 muestra la evolución de los ordenadores más rápidos del mundo.

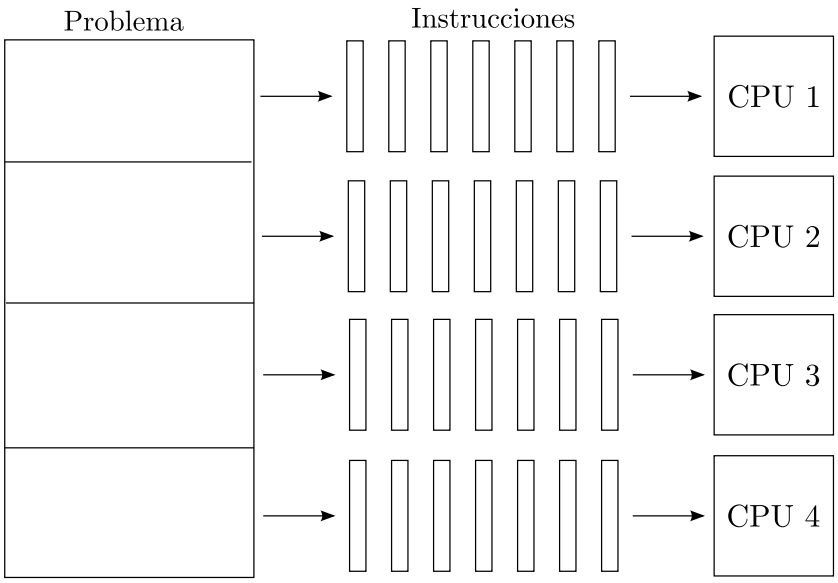


Figura 1.8: Esquema de ejecución de un programa en parallelo.

1.5.1 ¿Cuándo es necesario?

Los motivos clásicos más importantes para utilizar el cálculo en paralelo son:

- Resultados en menos tiempo.
- Resolución de problemas más grandes en memoria y/o en operaciones.

Además, hoy en día las arquitecturas de los procesadores son de n-núcleos y para sacarles todo el rendimiento es necesario hacer uso del cálculo en paralelo.

1.5.2 Paradigmas de programación en paralelo

La clasificación más habitual de los ordenadores paralelos es atendiendo a la distribución de memoria:

- Ordenadores de memoria compartida: todas las CPUs acceden a la misma memoria. (paradigma OpenMP)
- Ordenadores de memoria distribuida: cada CPU tiene su propia memoria local que no es visible por el resto de CPUs. La información es compartida por una red. (paradigma MPI).
- Cálculo en GPUS + CPU. (paradigma GPU)

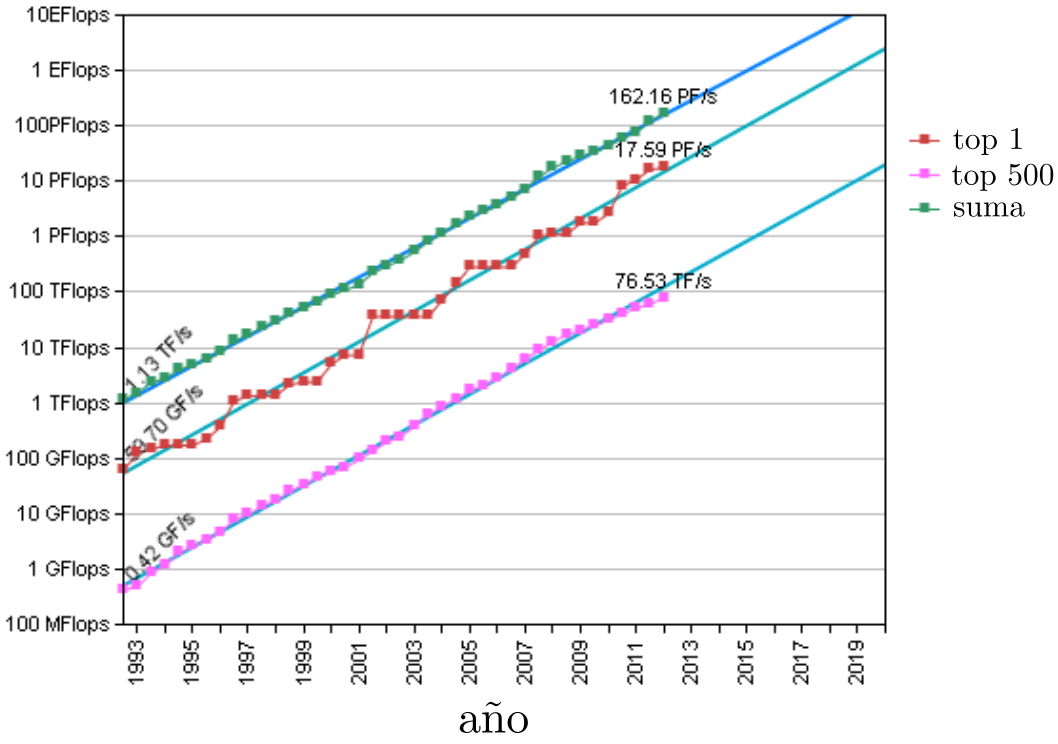


Figura 1.9: Evolución de los ordenadores más potentes del mundo. Fuente: *http://www.top500.org* .

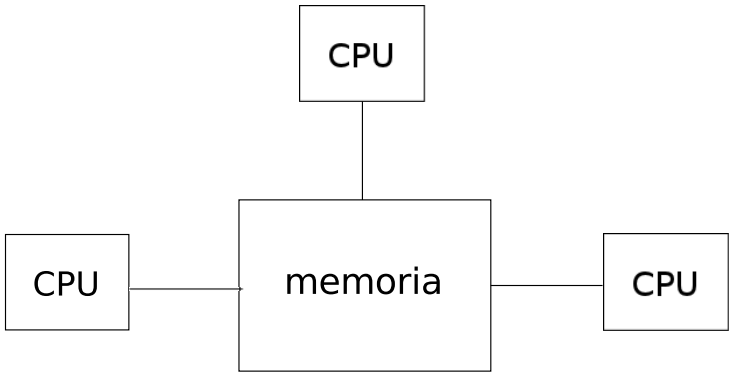


Figura 1.10: Paradigmas de cálculo en paralelo. Memoria compartida.

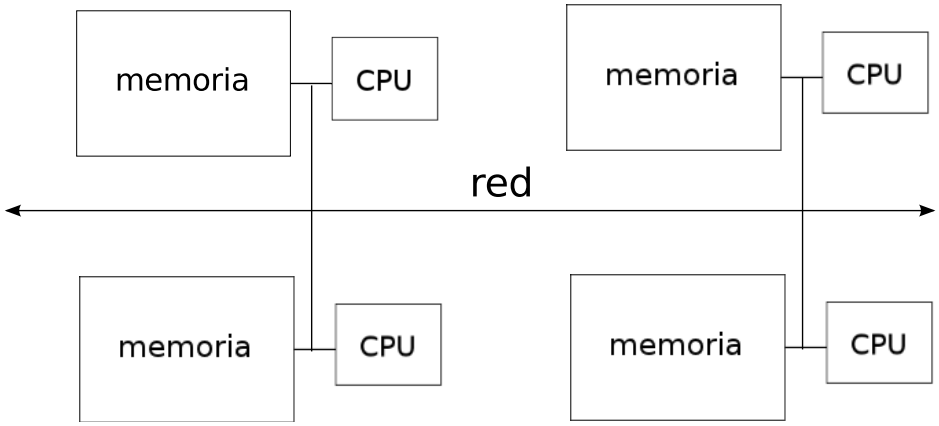


Figura 1.11: Paradigmas de cálculo en paralelo. Memoria distribuida.

- Ordenadores híbridos. Grupos de CPUs comparten la misma memoria (y tal vez GPU) y se comunican con otros grupos a través de una red.

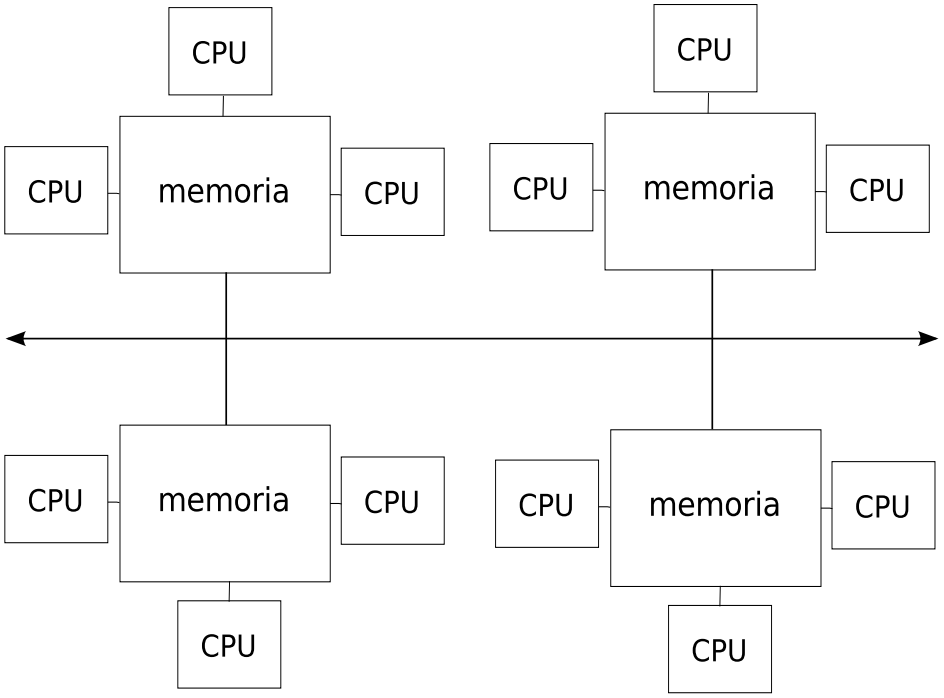


Figura 1.12: Paradigmas de cálculo en paralelo. Híbrido de memoria compartida + distribuida.

Capítulo 2

Planteamiento del problema CFD

El punto de inicio de todo método numérico es el modelo matemático del fenómeno físico que se desea estudiar y que generalmente suele ser expresado en forma de ecuaciones diferenciales en derivadas parciales o ecuaciones integro-diferenciales junto con las condiciones de contorno. En el caso de la dinámica de fluidos computacional se utilizan las ecuaciones de Navier-Stokes o simplificaciones de las mismas dependiendo de la aplicación.
Como ya hemos mencionado en el capítulo anterior, el ordenador es una máquina finita y no puede manejar ecuaciones en derivadas parciales con variables continuas en el espacio y el tiempo. Por ello, una vez definido el problema matemático que se quiere resolver, se procede a realizar la discretización temporal y espacial, transformando las ecuaciones diferenciasles en algebraicas. La solución que obtenemos no será continua sino que vendrá dada por una serie discreta de valores tanto en el espacio como en el tiempo.

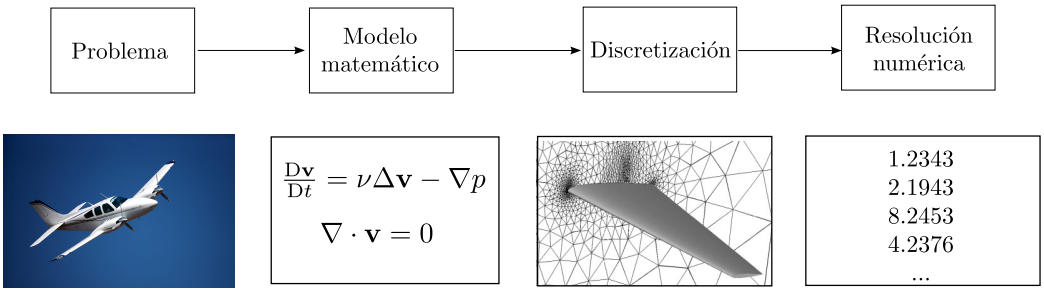


Figura 2.1: Pasos para resolver numéricamente un problema con CFD.

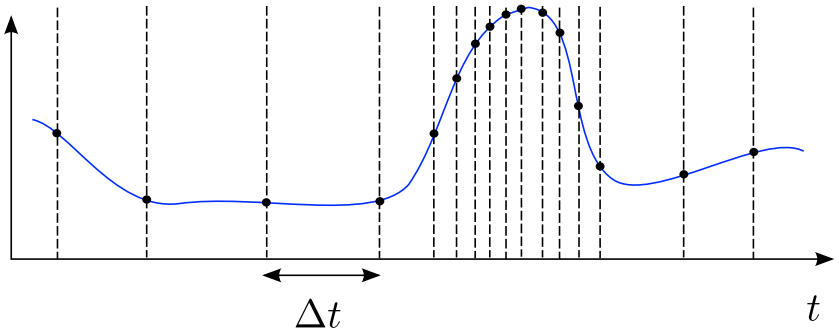


Figura 2.2: Discretización temporal. El paso de tiempo debe ser el adecuado para captar los cambios de la solución.

2.1 Ideas generales de la discretización temporal

En el cálculo de flujos no estacionarios debemos discretizar la coordenada temporal. La solución se obtiene en puntos discretos del tiempo tal y como muestra la figura 2.2. El tiempo transcurrido entre dos instantes define el paso de tiempo Δt. Un aspecto importante a la hora de usar Δt es que éste debe ser tal que capte los cambios rápidos de la solución. La principal diferencia entre espacio y tiempo recae en la dirección de influencia: mientras que una fuerza puede influenciar todos los puntos del espacio (en problemas elípticos) esa misma fuerza al ser aplicada en un instante dado sólo puede afectar al futuro. Los flujos no estacionarios tienen carácter parabólico. Por ello, la mayor parte de los métodos numéricos para resolver la coordenada espacial se basan en avanzar paso a paso en el tiempo.

2.2 Ideas generales de la discretización espacial

Tanto en los flujos estacionarios como no estacionarios se debe proceder a la discretización espacial para obtener la solución numérica. Las posiciones discretas en las que las variables son calculadas están definidas por la malla numérica, que es esencialmente una representación discreta del dominio geométrico del problema. La malla divide el dominio en un número finito de subdominios (elementos, volúmenes de control, nodos...). El mallado espacial presenta mayor complejidad que el temporal, debido a que tenemos tres dimensiones, el dominio puede ser de geometría compleja y ademas es difícil

predecir a priori en qué lugares va a ser necesario un mallado más fino.

2.2.1 Clasificación de métodos de discretización espacial

Los principales métodos de discretización espacial están asociados a las diferentes formulaciones del problema matemático: forma diferencial, integral o débil.

- **Métodos de diferencias finitas**
  Utilizan la formulación diferencial de las ecuaciones. El dominio se cubre con puntos llamados nodos en los cuales la ecuación es aproximada remplazando las derivadas parciales por aproximaciones en términos de los valores nodales de la función. Cuando se aplican en mallas estructuradas (ver siguiente apartado) son muy sencillos y efectivos. Además es fácil obtener esquemas de alto orden. Entre sus inconvenientes están que la conservación de momento y masa no está garantizada si no se tiene especial cuidado y es complicada su aplicación a dominios de geometrías irregulares.

- **Métodos de volúmenes finitos**
  Utilizan la formulación integral de las ecuaciones. El dominio se divide en volúmenes de control en los cuales se aplican las ecuaciones integrales que son aproximadas mediante cuadraturas. En este caso los nodos residen en el centroide del volumen y se interpolan para obtener sus valores en las caras de dichos volúmenes. Se pueden usar cómodamente en todo tipo de mallas, tanto estructuradas como no estructuradas (ver siguiente sección). Otra de sus ventajas es que son conservativos por construcción y todos los términos aproximados tienen un sentido físico claro. Entre sus desventajas está la dificultad de obtener esquemas de alto orden, sobre todo en 3D, debido a que requieren tres niveles de aproximación: interpolación, diferenciación e integración. Es el método utilizado por la mayoría de software CFD (ANSYS FLUENT, STAR_CCM+, OPENFOAM...)

- **Métodos de elementos finitos**
  Utilizan la formulación débil: la ecuación diferencial es multiplicada por unas funciones llamadas *pesos* y posteriormente integradas. Son similares en cierto modo al método de volúmenes finitos. El dominio se divide en elementos y en cada uno de ellos la solución es aproximada, generalmente de forma lineal, utilizando los valores de la función en los

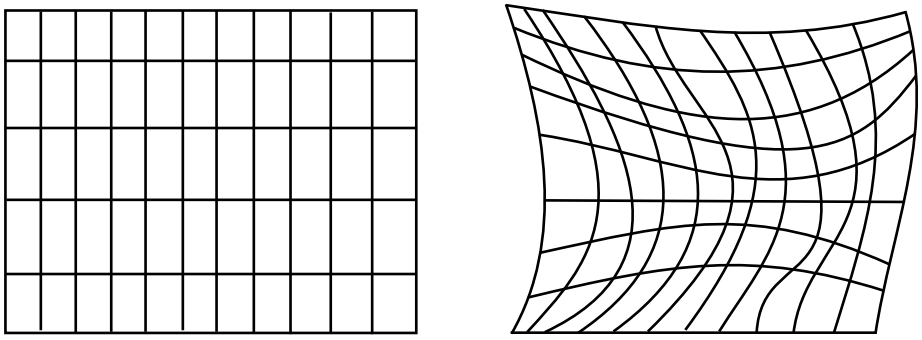


Figura 2.3: Ejemplo de mallas estructuradas.

vértices del elemento. Esta aproximación es sustituida en la ecuación integral pesada y se impone que la derivada de dicha integral con respecto al valor en cada nodo sea cero. Son apropiados para geometrías complejas y fáciles de analizar matemáticamente. Menos común en CFD pero también se pueden encontrar paquetes de software como ELMER, FENICS...

- Otros: métodos espectrales, método paneles...

2.2.2 Clasificación de mallas

- **Mallas estructuradas.**
  Las mallas estructuradas son aquellas formadas por un conjunto de nodos (o volúmenes de control) que pueden ser identificados de forma única mediante un grupo de índices ordenados (i, j, k) en 3D ó (i, j) en 2D. Es el tipo de malla más simple y es equivalente a una malla cartesiana mediante el cambio de coordenadas apropiado. Cada nodo P de la malla tiene 4 vecinos en 2D y 6 en 3D a los cuales se accede variando los indices (i, j, k) de P en ± 1. Su mayor desventaja es que sólo pueden ser utilizadas en dominios con geométrías simples y muchas veces acumulan puntos en regiones que no son de interés. Suelen ser las mallas más utilizadas en los métodos de elementos finitos. Gran cantidad de algoritmos están diseñados para mallas cartesianas regulares y son aplicados a otras mallas mediante una transformación de coordenadas.

  Las mallas estructuradas se subdividen a su vez en tres grupos según cómo sea la deformación que hay que aplicar a una malla cartesiana para obtenerlas: mallas tipo O, tipo C ó tipo H. En una malla tipo O tenemos puntos organizados circularmente de tal forma que las líneas

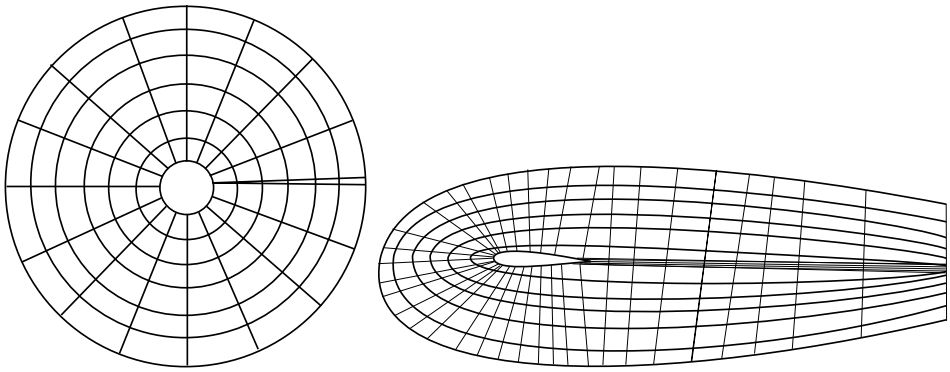


Figura 2.4: Ejemplos de mallas estructuradas tipo O y tipo C.

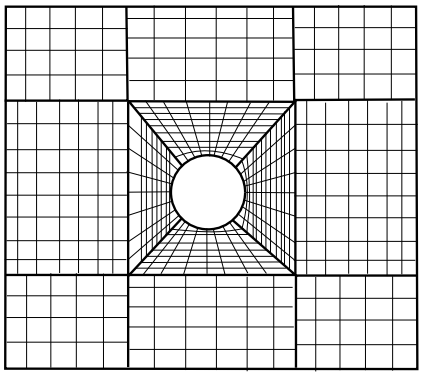


Figura 2.5: Ejemplo de malla estructurada multi-bloque.

que los unen son cerradas, y por lo tanto, parecen una O. En las mallas tipo C las lineas se doblan reproduciendo la forma de C. Al resto de mallas se las denomina tipo H.

  - **Mallas estructuradas multi-bloque.**
    En las mallas estructuradas multi-bloque hay uno o más niveles de subdivisión. En el nivel exterior, hay bloques generalmente grandes que pueden ser de estructura irregular e incluso solaparse. En el nivel más fino se definen mallas estructuradas con un tratamiento especial de las regiones de acoplamiento entre bloques. Este tipo de mallas es más flexible que las estructuradas y permite usar mayor resolución en aquellas regiones donde es necesario, aunque son más complejas de programar.

- **Mallas no-estructuradas.**
  Para geometrías muy complejas, las mallas más flexibles son aquellas

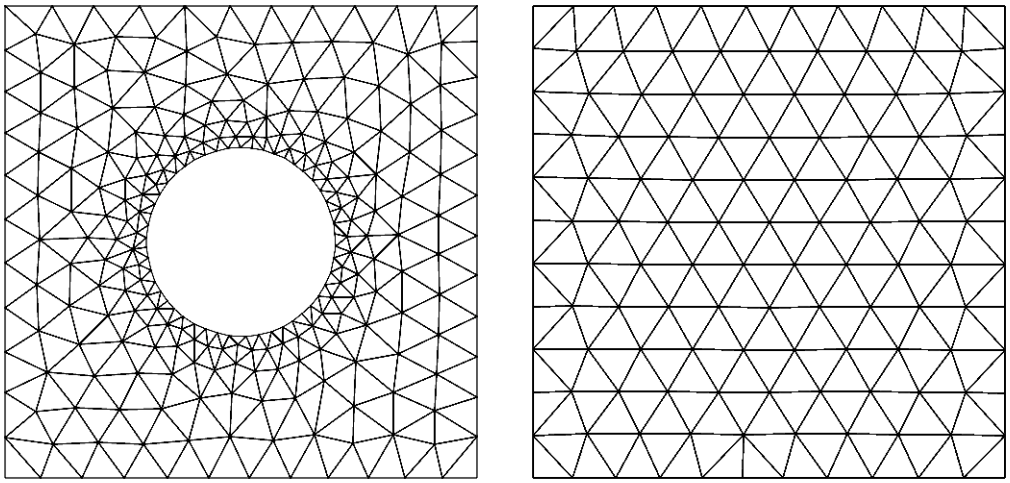


Figura 2.6: Ejemplos de mallas no-estructuradas. Tomadas de *Distmesh. A Simple Mesh Generator in MATLAB©*.

que se pueden adaptar de forma arbitraria al dominio. En principio, este tipo de mallas pueden ser usadas con cualquier esquema de discretización espacial, sin embargo, los métodos de volúmenes y elementos finitos son los que mejor se adaptan. Los elementos o volúmenes de control pueden tener cualquier forma, sin restricciones en cuanto al número de elementos vecinos ni nodos. En la práctica, las mallas se construyen utilizando triángulos o cuadriláteros en 2D y tetraedros o hexaedros en 3D. Existe una gran variedad de trabajos dedicados al estudio de la generación de mallas no-estructuradas de forma automática. La ventaja de su flexibilidad contrasta con la estructura irregular de los datos que produce y la necesidad de usar algoritmos más complicados y caros ya que las matrices que hay que resolver son llenas.

- **Mallas híbridas.**
  En algunos casos se combinan los diferentes tipos de malla expuestos anteriormente. En estos casos hay que tener cuidado con el acoplamiento en las diferentes mallas.

2.2.3 Generación de mallas

En la mayoría de la literatura se establece como primer criterio de clasificación de mallas el tipo de malla creada y, en segundo lugar, el modo en el que se genera. Siguiendo estas pautas, las distintas técnicas de discretización

se pueden dividir en:

- **Métodos de generación de malla estructurada:**
  - Métodos algebraicos: se obtienen aplicando una transformación de coordenadas a geometrías canónicas simples (mapping).
  - Métodos basados en EDPs: Basados en la resolución de EDPs (generalmente elípticas), con condición de contorno la geometría del contorno del dominio que se pretende discretizar. Similares a los métodos algebraicos pero las coordenadas de los nodos interiores vienen determinadas por la resolución de estas EDPs. Presentan alto coste computacional comparados con los métodos algebraicos.
- **Métodos de generación de malla no estructurada:**
  - Método de Delaunay-Voronoï: Primero colocamos en el dominio los nodos en los lugares deseados (lo cual puede ser no trivial), y obtenemos un conjunto de puntos P_i. Dado ese conjunto de puntos, se pueden definir unas regiones poliédricas V_i asociadas a cada punto, de modo que cualquier punto de la región V_i se encuentra más cerca al punto P_i que a cualquiera del resto. Cada una de estas regiones se denomina región de Voronoï . A partir de su definición resulta evidente que cada cara de estas regiones poliédricas se encuentra equidistante de los dos puntos que separa. La unión de todos estos puntos por pares genera otra discretización del dominio, conocida como triangulación de Delaunay, que posee una característica muy interesante para la generación de mallas: la regularidad de ángulos en los triángulos generados es máxima. Es decir, dado un conjunto de nodos, el método de Delaunay garantiza una triangulación óptima. Sin embargo, en el caso volumétrico, esta triangulación óptima no garantiza que los tetraedros generados sean óptimos, por lo que, en general, tras la generación de la malla son necesarias técnicas de detección y corrección de tetraedros defectuosos.
  - Método de frente de avance: se realiza desde el contorno hacia el interior del dominio. Se analiza un frente, inicializado con los datos del contorno, para determinar una zona de partida desde la que se crean uno o varios elementos internos, junto con los correspondientes nodos y aristas. Seguidamente se actualiza el frente con los nuevos nodos y aristas generadas y se repite el proceso hasta que el dominio queda completamente mallado.

– Métodos Multibloque: la idea consiste en la división del dominio en bloques de topología más sencilla, cada bloque se procesa posteriormente con alguna de las técnicas anteriores.

Capítulo 3

Discretización temporal

3.1 Problema de condiciones iniciales

La discretización temporal se aplica a problemas de evolución definidos por ecuaciones diferenciales ordinarias de primer orden en el tiempo junto con las condiciones iniciales correspondientes. A este tipo de problemas se les denomina *problemas de Cauchy* y son de la forma

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= F(u,t), \\ u(t_0) &= u_0, \end{aligned} \tag{3.1}$$

donde t es la variable independiente, u y F vectores columna de dimensión s y u_0 la condición inicial. Aunque no es habitual que aparezcan derivadas de más de segundo orden en el tiempo, cualquier sistema se puede reducir a primer orden realizando un cambio de variable. Partiendo del sistema de dimensión uno y orden s

$$\frac{\mathrm{d}^s y}{\mathrm{d}t^s} = F(y, \frac{\mathrm{d}y}{\mathrm{d}t}, ..., \frac{\mathrm{d}^{s-1} y}{\mathrm{d}t^{s-1}}, t) \tag{3.2}$$

lo podemos reducir a dimensión s y orden uno tomando $u_1 = y$, $u_2 = \mathrm{d}y/\mathrm{d}t$,...,$u_s = \mathrm{d}^{s-1}y/\mathrm{d}t^{s-1}$, dando como resultado

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = u_{i+1}, \quad i = 1, ..., s-1, \tag{3.3}$$

$$\frac{\mathrm{d}u_s}{\mathrm{d}t} = F(u_1, ..., u_s, t). \tag{3.4}$$

La idea de la discretización temporal es transformar la ecuación diferencial (3.1) en una ecuación algebraica (ecuación en diferencias) que podamos re-

solver con un ordenador. Como resultado, obtendremos los valores aproximados de $u(t)$ en una serie discreta de puntos en el tiempo, t_n. A continuación pasamos a describir la nomenclatura:

- $u(t)$ es la solución exacta de la ecuación (3.1), donde ambas u y t son variables continuas.
- u_0 es la condición inicial en el instante $t = t_0$.
- t_n con $n = 1, ..., N$ son los valores discretos de t donde obtendremos la aproximación numérica a la función $u(t)$. Llamaremos paso de tiempo a $\Delta t_n = t_{n+1} - t_n$, que en general dependerá de n. En muchas ocasiones consideraremos que el paso de tiempo es constante y lo llamaremos simplemente Δt.
- $u(t_n)$ es la solución exacta evaluada en el instante $t = t_n$.
- u^n es la aproximación numérica a la solución exacta $u(t_n)$ en el instante t_n. En general $u^n \neq u(t_n)$.
- $F^n = F(u^n, t_n)$ es la evaluación de F con la aproximación numérica en el instante t_n. En general $F^n \neq F(u(t_n), t_n)$.
- Expresaremos un esquema numérico genérico de la forma:

$$\sum_{j=0}^{p} \alpha_j u^{n+1-j} = \Delta t H(u^{n+1}, ..., u^{n+1-p}, t_n, ...), \tag{3.5}$$

  con $j = 0, 1, .., p$, donde p es el número de pasos y H una función que depende de $F(u, t)$ y del esquema.
- Error local de truncación: $T^n = o(\Delta t^{q+1})$ con q el orden del esquema numérico.
- Error global: $E^n = u(t_n) - u^n = o(\Delta t^q)$, con q el orden del esquema numérico.

3.2 Clasificación de esquemas numéricos

Podemos realizar dos grandes clasificaciones de los esquemas numéricos atendiendo bien al sistema de ecuaciones que hay que resolver o bien al número de instantes implicados para obtener la solución en cada paso temporal. Consideraremos un esquema numérico como el dado por la ecuación (3.5).

- **Esquemas numéricos unipaso, multipaso, multietapa.**
  Unipaso:
  Sólo involucran un paso de tiempo anterior a t_{n+1}, que denominaremos u^{n-l} y el que se quiere calcular, u^{n+1}. Son de la forma

$$u^{n+1} = u^{n-l} + \Delta t H(u^{n+1}, u^{n-l}, t_{n+1}, t_{n-l}), \tag{3.6}$$

  con l fijo y generalmente $l = 0$ ó $l = 1$. Entre sus ventajas está su ahorro de memoria, puesto que sólo es necesario almacenar la solución en un único instante anterior. Además aquellos con $l = 0$ no presentan soluciones espúreas. Ejemplos: esquemas Euler explícito e implícito, Crank-Nicolson.

  Multipaso:
  La solución en el instante t_{n+1} se obtiene usando la información de p instantes anteriores t_{n-j+1} con $j = 1, ..., p$. Se dice entonces que es un esquema de p pasos. Son de la forma

$$u^{n+1} = -\sum_{j=1}^{p} \alpha_j u^{n-j+1} + \Delta t H(u^{n+1}, ..., u^{n-p+1}, t_{n+1}, ..., t_{n-p+1}). \tag{3.7}$$

  Presentan como inconveniente que es necesario almacenar en memoria p instantes anteriores lo cual es inasumible en problemas grandes. Además, necesitamos p valores iniciales para arrancarlos cuando en principio sólo contamos con $u^0 = u_0$, por lo que se suelen arrancar de forma escalonada usando esquemas de menos pasos. Otro problema importante son las soluciones espúreas que aparecen y que es necesario controlar para descartarlas. Entre sus ventajas está alcanzar mayor orden que los esquemas unipaso (que no sea multietapa) al usar más información. Ejemplos: esquemas Adams (Bashforth y Moulton) con $p = 1$.

  Multietapa:
  Los esquemas numéricos multietapa son aquellos en los que se halla la solución iterativamente usando varias etapas. Utilizan no sólo los instantes t_n y t_{n+1} sino también otros intermedios. Suelen ser unipaso, aunque teóricamente también pueden ser multipaso. Tienen grandes ventajas tales como la ausencia de soluciones espúreas, alto orden y estabilidad sin necesidad de tanta memoria como los multipaso. Ejemplo: esquemas Runge-Kutta.

- **Esquemas numéricos explícitos o implícitos.**
  Explícitos:
  Son aquellos esquemas en los que para calcular u^{n+1} se utilizan valores conocidos en instantes anteriores u^{n-j+1} con $j = 1, ..., p$.

$$u^{n+1} = -\sum_{j=1}^{p} \alpha_j u^{n-j+1} + \Delta t H(u^n, ..., u^{n-p+1}, t_n, ..., t_{n-p+1}). \quad (3.8)$$

  Son sencillos de programar dado que no es necesario resolver ningún sistema de ecuaciones algebraicas no lineal, sino que la solución se obtiene directamente evaluando $H(u^n, ..., u^{n-p+1})$, que no depende de u^{n+1}. Su principal desventaja es que son inestables para Δt grandes. Ejemplos: Euler explícito, Leap-Frog, Adams-Bashforth, predictor-corrector, Runge-Kutta explícitos.

  Implícitos:
  Aquellos en los que para calcular u^{n+1} se utilizan valores conocidos en instantes anteriores u^{n-j+1} con $j = 1, ..., p$ junto con u^{n+1},

$$u^{n+1} = -\sum_{j=0}^{p} \alpha_j u^{n-j+1} + \Delta t H(u^{n+1}, ..., u^{n-p+1}, t_n, ..., t_{n-p+1}). \quad (3.9)$$

  Son complejos de programar y la solución es más cara de obtener ya que es necesario resolver un sistema de ecuaciones algebraicas no lineales. Entre sus principales ventajas están su estabilidad para Δt grandes en comparación con su equivalente explícito, que hace que se pueda avanzar la solución más rápidamente por paso. Ejemplos: Euler implícito, Crank-Nicolson, Adams-Moulton, Runge-Kutta implícito.

3.3 Obtención de esquemas numéricos

Existen dos métodos básicos para la obtención de esquemas numéricos: la *cuadratura numérica* y la *diferenciación numérica*. Muchos esquemas se pueden obtener usando tanto un método como el otro. Existen además otros esquemas numéricos que se construyen combinando los anteriores.

- **Cuadratura numérica.**
  En la cuadratura numérica el problema (3.1) es integrado entre t_n y t_{n+1} para obtener

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} F(u, t)\mathrm{d}t. \quad (3.10)$$

La relación anterior es exacta y los diferentes esquemas numéricos se obtienen al aproximar la integral

$$\int_{t_n}^{t_{n+1}} F(u,t)\mathrm{d}t. \tag{3.11}$$

Existen diferentes formas de hacerlo. Una de ellas consiste en utilizar el desarrollo en serie de Taylor de $F(u(t),t)$. Otra, la que usaremos nosotros, se basa en definir una función de interpolación para $F(u(t),t)$ e integrarla entre t_n y t_{n+1}. La integración se hace entre los puntos t_n y t_{n+1}, sin embargo, para construir la interpolación podemos usar no sólo esos puntos sino también otros intermedios (esquemas multietapa) o anteriores como t_{n-1}, t_{n-2}... (esquemas multipaso). Hay que tener en cuenta que t es la única variable independiente y tanto los desarrollos en serie de Taylor como la interpolación se hacen en el tiempo. La figura 3.1 muestra varios esquemas numéricos obtenidos con diferentes aproximaciones del área bajo $F(u,t)$. En general, utilizaremos un polinomio interpolante[1] para $F(u,t)$ de la forma

$$F(u,t) \approx \sum_{j=n-p+1}^{n+1} F^j L_j(t), \tag{3.12}$$

donde F^j son valores de $F(u,t)$ en los instantes usados para interpolar u_j y t_j $j = n-p+1, ..., n+1$ y $L_j(t)$ son las funciones de interpolación, por ejemplo, los polinomios de Lagrange. El esquema numérico se obtiene introduciendo la aproximación (3.12) en la ecuación (3.10) y sustituyendo $u(t_n)$ y $u(t_{n+1})$ por u^{n+1} y u^n dado que ya no manejamos la solución exacta sino su aproximación númerica. El resultado es

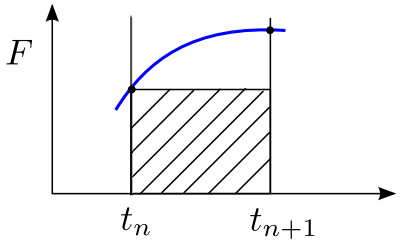
$$u^{n+1} = u^n + \int_{t_n}^{t_{n+1}} \sum_{j=n-p+1}^{n+1} F^j L_j(t)\mathrm{d}t. \tag{3.13}$$

que se puede expresar de la forma

$$u^{n+1} = u^n + \Delta t \sum_{j=0}^{p} \beta_j F^{n-j+1}. \tag{3.14}$$

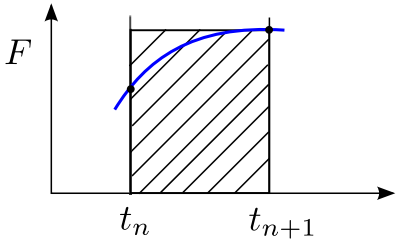
y da lugar a esquemas como los que se muestran en la figura 3.2 denominados Adams-Bashforth cuando $\beta_0 = 0$ y Adams-Moulton cuando $\beta_0 \neq 0$.

Euler explícito



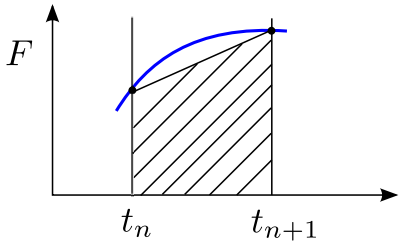
$$\begin{aligned} u^{n+1} &= u^n + \Delta t F(u^n, t_n) \\ E^n &= o(\Delta t) \end{aligned}$$

Euler implícito



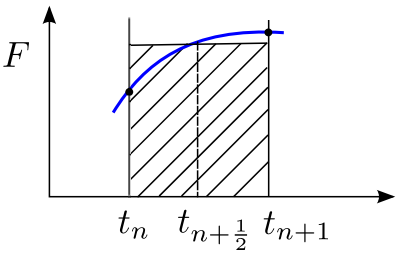
$$\begin{aligned} u^{n+1} &= u^n + \Delta t F(u^{n+1}, t_{n+1}) \\ E^n &= o(\Delta t) \end{aligned}$$

Crank-Nicolson



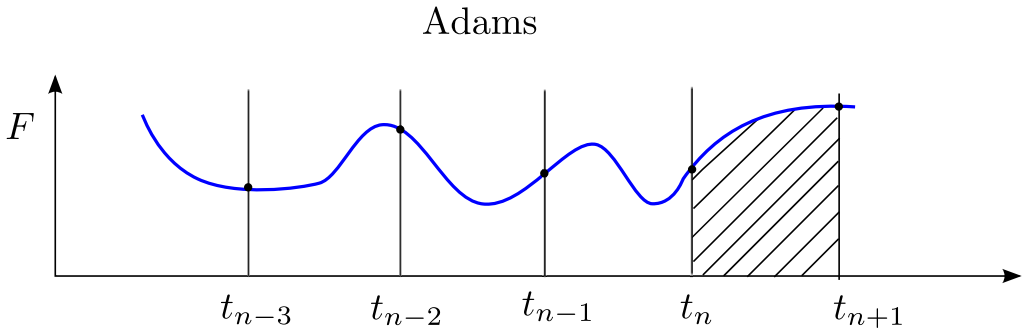
$$\begin{aligned} u^{n+1} &= u^n + \frac{\Delta t}{2}\left(F^n + F^{n+1}\right) \\ E^n &= o(\Delta t^2) \end{aligned}$$

Leap-frog



$$\begin{aligned} u^{n+1} &= u^n + \Delta t F(u^{n+\frac{1}{2}}) \\ E^n &= o(\Delta t^2) \end{aligned}$$

Figura 3.1:



$$u^{n+1} = u^n + \Delta t \sum_{j=0}^{p} \beta_j F^{n+1-j}$$

$\beta_0 = 0 \rightarrow$ Adams-Bashforth $\qquad$ $\beta_0 \neq 0 \rightarrow$ Adams-Moulton

$$E^n = o(\Delta t^p) \qquad\qquad E^n = o(\Delta t^{p+1})$$

Figura 3.2:

Algunos esquemas Adams-Bashforth:

- Primer orden: $u^{n+1} = u^n + \Delta t F^n$ (Euler explícito).
- Segundo orden: $u^{n+1} = u^n + \Delta t/2\,(3F^n - F^{n-1})$.

Algunos esquemas Adams-Moulton:

- Primer orden: $u^{n+1} = u^n + \Delta t F^{n+1}$ (Euler implícito).
- Tercer orden: $u^{n+1} = u^n + \Delta t/12\,(5F^{n+1} + 8F^n - F^{n-1})$.

- **Diferenciación numérica.**
  En la diferenciación numérica usamos la ecuación original

$$\frac{\mathrm{d}u}{\mathrm{d}t} = F(u,t), \tag{3.15}$$

y aproximamos la derivada temporal $\mathrm{d}u/\mathrm{d}t$. Para ello, calculamos una función de interpolación de $u(t)$ a partir de su valor en los instantes

[1]El polinomio interpolante de Lagrange de u en un conjunto de puntos $(u_0,t_0),...,(u_n,t_n)$ viene dado por $\sum_{j=0}^{n} u_j L_j(t)$ con $L_j(t) = \prod_{i=0,i\neq j}^{n} \frac{t-t_i}{t_j-t_i}$. Si utilizamos $n+1$ puntos el error cometido será del orden Δt^{n+1}.

t_{n+1}, t_n, t_{n-1}... la derivamos y obligamos a que se satisfaga en t_n ó t_{n+1}. En general, utilizaremos un polinomio interpolante para u de la forma

$$u(t) \approx \sum_{j=n-p+1}^{n+1} u^j L_j(t), \tag{3.16}$$

lo derivamos para obtener

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=n-p+1}^{n+1} u^j L_j(t) = F(u,t), \tag{3.17}$$

y lo particularizamos en t_{n+1} o t_n para obtener esquemas de la forma

$$\sum_{j=0}^{p} \alpha_j u^{n-j+1} = \Delta t F^k, \tag{3.18}$$

con $k = n$ o $k = n+1$. Por ejemplo, el polinomio interpolante de Lagrange de $u(t)$ usando los dos puntos t_n y t_{n+1} puede expresarse de la forma

$$u(t) \approx u(t_n)\frac{t-t_{n+1}}{t_n-t_{n+1}} + u(t_{n+1})\frac{t-t_n}{t_{n+1}-t_n}, \quad \forall t \in [t_n, t_{n+1}] \tag{3.19}$$

suponiendo un paso de tiempo constante Δt, la primera derivada de $u(t)$ puede aproximarse por

$$\frac{\mathrm{d}u}{\mathrm{d}t} \approx \frac{u(t_{n+1})-u(t_n)}{\Delta t}, \tag{3.20}$$

lo que nos permite aproximar la ecuación diferencial como

$$\frac{u(t_{n+1})-u(t_n)}{\Delta t} \approx F(u,t). \tag{3.21}$$

Particularizando la expresión en $t = t_n$ obtenemos la expresión del Euler explícito

$$u^{n+1} = u^n + \Delta t F(u^n, t_n). \tag{3.22}$$

Si por el contrario particularizamos en el instante $t = t_{n+1}$ obtenemos el esquema Euler implícito

$$u^{n+1} = u^n + \Delta t F(u^{n+1}, t_{n+1}). \tag{3.23}$$

Sumando las dos expresiones (3.22) y (3.23) anteriores y multiplicando la primera por $(1-\theta)$ y la segunda por θ con $0 \leq \theta \leq 1$ se obtiene la familia de los θ-métodos.

También se puede utilizar el desarrollo en serie de Taylor de $u(t)$ en lugar de una función de interpolación para obtener esquemas por diferenciación numérica.

- **Otros métodos: predictor-corrector.**
  La idea de los métodos predictor-corrector consiste en hacer una estimación de la solución (predictor) con un esquema explícito para después corregirla (corrector) con un esquema implícito. Se combinan, por lo tanto, dos esquemas numéricos diferentes de tal forma que el esquema resultante sea explícito. En general, los pasos a seguir son:

  – Obtener una estimación de la solución u^{n+1} usando el esquema explícito predictor: u_*^{n+1}.

$$\alpha_0 u_*^{n+1} + \sum_{j=1}^{p} \alpha_j u^{n+1-j} = \Delta t H_e(u^n, u^{n-1}, ..., t_n, ...), \qquad (3.24)$$

    donde H_e no depende de u^{n+1}.

  – Utilizar un esquema implícito corrector para obtener la solución definitiva utilizando u_*^{n+1} en vez de u^{n+1} en H_i y convertir así el esquema en explícito.

$$\gamma_0 u^{n+1} + \sum_{j=1}^{p} \gamma_j u^{n+1-j} = \Delta t H_i(u_*^{n+1}, u^n, u^{n-1}, ..., t_n, ...). \quad (3.25)$$

  A veces el proceso anterior es más largo y se itera varias veces hasta obtener el error deseado. La ventaja que presentan los esquemas predictor-corrector reside en aumentar el orden de un esquema explícito sin aumentar mucho el coste computacional. Es común construir parejas predictor-corrector usando esquemas Adams-Bashforth de orden q ó $q-1$ como predictor y Adams-Moulton de orden q como corrector.

- **Otros métodos: Runge-Kutta.**
  La forma general de los esquemas Runge-Kutta está recogida en la figura 3.3 y es

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{e} b_i k_i, \qquad (3.26)$$

$$k_i = F(u^n + \Delta t \sum_{j=1}^{e} a_{ij} k_j, t_n + c_i \Delta t), \quad i = 1, ..., e. \quad (3.27)$$

  Se basan en la idea de estimar la función F en pasos intermedios denominados etapas. Se pueden entender como esquemas predictor-corrector o como un proceso iterativo en el que en cada etapa se mejora la estimación anterior.

Los coeficientes de los esquemas Runge-Kutta se suelen organizar usando la *tabla de Butcher*.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1e} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2e} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_e & a_{e1} & a_{e2} & \cdots & a_{ee} \\ \hline & b_1 & b_2 & \cdots & b_e \end{array} \tag{3.28}$$

Que se puede expresar como

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \tag{3.29}$$

Si la matriz A es triangular inferior estricta el método es explícito y en caso contrario es implícito. Para obtener los coeficientes del esquema se desarrolla en serie de Taylor la expresión (3.26) y se iguala al desarrollo de $\mathrm{d}u/\mathrm{d}t$.

Algunos esquemas Runge-Kutta:

- Segundo orden:

$$\begin{aligned} u^{n+1} &= u^n + 1/2\,(k_1 + k_2)\,, \\ k_1 &= \Delta t F^n, \\ k_2 &= \Delta t F(u^n + k_1, t_n + \Delta t). \end{aligned}$$

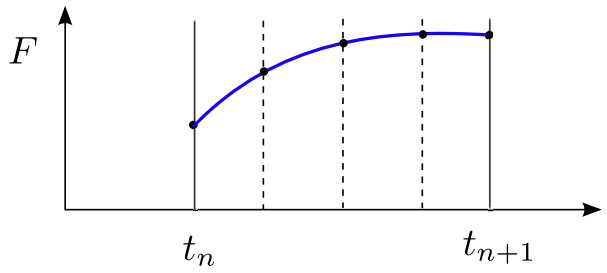
- Tercer orden:

$$\begin{aligned} u^{n+1} &= u^n + 1/6\,(k_1 + 4k_2 + k_3)\,, \\ k_1 &= \Delta t F^n, \\ k_2 &= \Delta t F(u^n + k_1/2, t_n + \Delta t/2), \\ k_3 &= \Delta t F(u^n - k_1 + 2k_2, t_n + \Delta t). \end{aligned}$$

- Cuarto orden (clásico):

$$\begin{aligned} u^{n+1} &= u^n + 1/6\,(k_1 + 2k_2 + 2k_3 + k_4)\,, \\ k_1 &= \Delta t F^n, \\ k_2 &= \Delta t F(u^n + k_1/2, t_n + \Delta t/2), \\ k_3 &= \Delta t F(u^n + k_2/2, t_n + \Delta t/2), \\ k_4 &= \Delta t F(u^n + k_3, t_n + \Delta t). \end{aligned}$$

Runge-Kutta



$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{e} b_i k_i$$

$$k_i = F(u^n + \Delta t \sum_{j=1}^{e} a_{ij} k_j, t_n + c_i \Delta t), \quad i = 1, ..., e$$

Explícito:

e	1	2	3	4	6	7	9
E^n	$o(\Delta t)$	$o(\Delta t^2)$	$o(\Delta t^3)$	$o(\Delta t^4)$	$o(\Delta t^5)$	$o(\Delta\ t^6)$	$o(\Delta t^7)$

Implícito: $E^n = o(\Delta t^{2e})$

Figura 3.3:

Los esquemas Runge-Kutta son sin duda esquemas de gran éxito, entre ellos el esquema RK4 clásico (Runge-Kutta orden 4). Entre sus ventajas están que sólo necesitan información de la solución en un paso, no presentan soluciones espúreas, pueden ser tanto explícitos como implícitos con gran estabilidad, permiten variar cómodamente el paso de tiempo y pueden alcanzar alto orden. Entre sus inconvenientes está la necesidad de evaluar varias veces la función F lo cual puede ser costoso.

3.4 Errores de la solución numérica

Para poder confiar en un resultado numérico es fundamental tener una estimación del error que se está cometiendo. Para realizar el estudio del error

consideraremos un esquema numérico genérico de la forma

$$\sum_{j=0}^{p} \alpha_j u^{n+1-j} = \Delta t H(u^{n+1}, ..., u^{n+1-p}, t_n, ...), \tag{3.30}$$

Podemos distinguir tres fuentes diferentes de error:

- **Error local de truncación**
  Es el asociado a cómo de buena es la aproximación del esquema numérico a la ecuación diferencial. Tal y como se vio en la sección anterior, los esquemas numéricos pueden obtenerse bien aproximando una cuadratura o aproximando la derivada temporal. En ambos casos es necesario truncar el desarrollo, es decir, usar un número finito de puntos de interpolación o de términos en el desarrollo en serie de Taylor, lo cual introduce inevitablemente un error. Si introducimos la solución exacta $u(t)$ en la ecuación $\mathrm{d}u/\mathrm{d}t = F(u,t)$, ésta se satisface. Sin embargo, ésto no ocurre si introducimos $u(t)$ en la ecuación del esquema numérico.

  Definición 1 *El error local de truncación de un esquema numérico en el instante t_{n+1} se define por*

  $$T^{n+1} = \sum_{j=0}^{p} \alpha_j u(t_{n+1-j}) - \Delta t H(u(t_{n+1}), ..., u(t_{n+1-p}), t_n, ...), \tag{3.31}$$

  donde $u(t_{n+1-j})$ es la solución exacta del problema de condiciones iniciales.

  Se puede demostrar que

  $$T^{n+1} \approx u(t_{n+1}) - \tilde{u}^{n+1}, \tag{3.32}$$

  donde $u(t_{n+1})$ es la solución exacta del problema de condiciones iniciales en t_{n+1} y $\tilde{u}^{n+1}$ es la solución numérica calculada partiendo de la solución exacta $u(t_n), u(t_{n-1}), ...$ y dando un paso. Al error de truncación también se le denomina *residuo* R^{n+1}.

- **Roundoff o Error de redondeo**
  Los ordenadores con los que se realizan los cálculos son máquinas finitas y las variables se representan con una precisión finita. Cada vez que el ordenador hace una operación trunca el resultado a 7 cifras significativas en el caso de simple precisión y a 15 en el caso de doble precisión. Para más detalles ver la sección §1.2.2.

- **Error de arranque de esquemas multipaso**
  Los esquemas multipaso de p pasos, necesitan ser arrancados con sucesivos esquemas de menos pasos lo cual introduce un error.

La acumulación en cada paso de los errores anteriores es lo que produce el *error global*.

Definición 2 *El error global de la solución numérica u^{n+1} en el instante t_{n+1} se define mediante*

$$E^{n+1} = u(t_{n+1}) - u^{n+1}, \tag{3.33}$$

donde $u(t_{n+1})$ es la solución exacta del problema de condiciones iniciales en t_{n+1} y u^{n+1} la solución aproximada con un esquema numérico partiendo de la condición inicial u^0 en t_0 y avanzando hasta t_{n+1}.

Un esquema numérico decimos que es de orden q si $E^{n+1} = o(\Delta t^q)$.
Estudiando la ecuación linealizada del error[2] se pueden obtener los siguientes resultados importantes:

- Si $T^{n+1} = o(\Delta t^{q+1})$ entonces $E^{n+1} = o(\Delta t^q)$.

- Los errores globales debidos a la pérdida de precisión están acotados por $o(\|\epsilon(t_n)\|)$ (epsilon de la máquina). Por ello, no tiene sentido coger un paso de tiempo Δt que produzca un error de truncación menor que la precisión de la máquina.

- No existe acumulación del error de las condiciones iniciales en los esquemas multipaso, por ello, un esquema de orden q se puede arrancar con un esquema de orden $q-1$.

Como normalmente no conocemos la solución exacta del problema, la **definición 2** no es muy útil. Para determinar el orden de un esquema numérico desarrollamos en serie de Taylor la expresión (3.31) y el error de truncación viene dado por la potencia del primer término en (Δt) distinto de cero. Una vez conocido que el error de truncación es de orden $q+1$, el error global será de orden q. Para los esquemas obtenidos usando un polinomio interpolante es fácil saber directamente cuál será su orden:

- Esquemas obtenidos por cuadraturas: Si utilizamos un polinomio de interpolación en m puntos para aproximar F, el error cometido será de orden Δt^m. Al integrar en el tiempo resulta un error de truncación de orden Δt^{m+1}.

[2]Para más detalles ver referencia [7]

- Esquemas obtenidos por diferenciación: Si utilizamos un polinomio de interpolación en m puntos para aproximar u, el error cometido será de orden Δt^m. Derivando y multiplicando por Δt para despejar la derivada, resulta un error de truncación de orden Δt^m.

3.5 Análisis de esquemas numéricos

3.5.1 Existencia y unicidad de la solución de la ecuación diferencial

Antes de buscar la solución numérica, es necesario estudiar la existencia, unicidad y estabilidad de la ecuación diferencial para saber si tiene sentido resolverla numéricamente y, en caso afirmativo, saber qué esquema numérico es más adecuado. Por ello, debemos resolver numéricamente aquellos problemas que denominamos *problemas bien planteados*.

Un problema bien planteado cumple:

- Existe solución.
- Es única.
- La solución varia regularmente con los parámetros (en caso de que los haya).

Generalmente los problemas mal planteados no representan de forma fidedigna la física del problema y deben ser reformulados. Para estudiar la existencia y unicidad de las soluciones del problema de condiciones iniciales

$$\frac{\mathrm{d}u}{\mathrm{d}t} = F(u,t), \tag{3.34}$$

$$u(t_0) = u_0, \tag{3.35}$$

disponemos del teorema de *Picard-Lindelöf* (o teorema de existencia y unicidad).

Teorema 1 *Sea $F(u,t)$, donde $F : \mathbb{R}^s \times \mathbb{R} \to \mathbb{R}^s$, definida y continua para todo (u,t) en la región*

$$\Omega = \{-\infty < u_i < \infty, i = 1, ..., s\} \times [t_0, t_f] \tag{3.36}$$

donde t_0 y t_f son finitos y sea una constante L tal que,

$$\|F(u,t) - F(u_*,t)\| \leq L\|u - u_*\| \tag{3.37}$$

se verifique para cada $(u,t),(u_*,t) \in \Omega$. *Entonces para cualquier* $u^0 \in \mathbb{R}^s$ *existe solución única al problema*

$$\frac{\mathrm{d}u}{\mathrm{d}t} = F(u,t), \tag{3.38}$$

$$u(t_0) = u_0, \tag{3.39}$$

donde $u(t)$ *es continua y diferenciable para todo* $(u,t) \in \Omega$.

La condición (3.37) es conocida como condición global de Lipschitz y quiere algo más que continuidad pero menos que diferenciabilidad. Por ello, desde el punto de vista práctico es suficiente comprobar que F es continua y que todas sus derivas parciales con respecto a u existen y son continuas (F de clase C^1) para garantizar la existencia y unicidad de la solución.

Teorema 2 *Si* $F(u,t)$ *es continua en* Ω *y existen y son continuas en* Ω *las derivadas* $\partial F/\partial u_i$, $i = 1,..,s$, *entonces existe solución única al problema de condiciones iniciales para todo* $(u^0,t_0) \in \Omega$.

Esta condición es más restrictiva pero más fácil de comprobar.

3.5.2 Estabilidad de la solución de la ecuación diferencial

Con el teorema de *Picard-Lindelöf* somos capaces de estudiar la existencia y unicidad de la solución. En caso de que tal solución exista, debemos estudiar a continuación su estabilidad. Nos interesa que el esquema numérico preserve al carácter de estabilidad de la solución, en concreto, nos interesa que si la solución de la ecuación diferencial es estable, la solución numérica también lo sea. Existen diferentes definiciones de estabilidad, aquí utilizaremos *estabilidad en sentido de Lyapunov.*

Teorema 3 *Sea* $u(t)$ *la solución única de (3.1) definida en* $[t_0,\infty)$. *Se dice que* $u(t)$ *es estable si para todo* $\epsilon > 0$, *existe* $\delta > 0$ *tal que la solución del problema de condiciones iniciales*

$$\frac{\mathrm{d}u_*}{\mathrm{d}t} = F(u_*,t), \quad u_*(t_0) = u_{0*}, \quad con \quad \|u_0 - u_{0*}\| < \delta \tag{3.40}$$

existe y está definida en $[t_0,\infty)$ *y verifica que* $\|u(t) - u_*(t)\| < \epsilon$ *para todo* $t \geq t_0$.

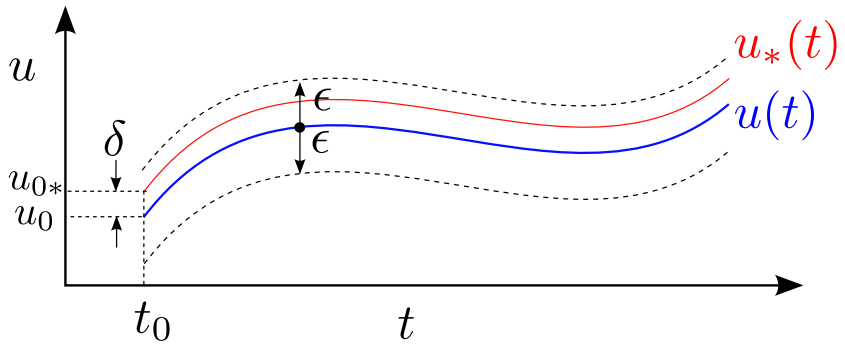


Figura 3.4: Interpretación de la estabilidad de una solución.

Si además la distancia $\|u(t) - u_*(t)\|$ tiende a cero con $t \to \infty$ se dice que es asintóticamente estable. La figura 3.4 muestra gráficamente la definición de estabilidad. Nótese que la estabilidad no es una propiedad de la ecuación diferencial sino de una solución concreta de la ecuación diferencial.
Podemos definir la solución $u_*(t) = u(t) + \Delta u(t)$, es decir, como la perturbación que hay que dar a $u(t)$ para obtener $u_*(t)$. Estudiar la estabilidad de una solución $u(t)$ puede llegar a ser extremadamente complicado en ecuaciones diferenciales no lineales y en vez de estudiar la solución de la ecuación no lineal se estudia la estabilidad de la ecuación linealizada. La ecuación linearizada que satisface Δu es

$$\frac{\mathrm{d}\Delta u}{\mathrm{d}t} = \frac{\partial}{\partial u} F(u,t)\Delta u + b(t) + N(\Delta u, t), \tag{3.41}$$

donde $N(\Delta u, t)$ contiene los términos no lineales. Cuando la solución $u(t) = u_0$ es constante o si el tiempo característico de variación del Jacobiano L es tal que lo podemos congelar en $u(t) = u_0$ y $t = t_0$, entonces podemos estudiar la estabilidad del sistema lineal. Por lo tanto, consideramos el sistema resultante de linealizar 3.1 en torno a una solución u con $\Delta u = u_L$ como

$$\frac{\mathrm{d}u_L}{\mathrm{d}t} = \frac{\partial}{\partial u} F(u_0, t_0)u_L + b(t). \tag{3.42}$$

El carácter de estabilidad de la solución del sistema anterior sólo depende de $\frac{\partial}{\partial u}F(u_0, t_0)$ y no del término $b(t)$[3], por lo que tenemos que analizar las estabilidad del sistema

$$\frac{\mathrm{d}u_L}{\mathrm{d}t} = Lu_L, \tag{3.43}$$

[3]Las soluciones de un sistema lineal de ecuaciones diferenciales ordinarias son de la forma $u(t) = \Phi(t)u_0 + \Phi(t)\int_{t_0}^{t} \Phi^{-1}(s)b(s)\mathrm{d}s$, con $\Phi(t)$ la matriz fundamental del sistema que cumple $\Phi(t_0) = I$. La estabilidad sólo depende de $\Phi(t)$ ya que el término $b(t)$ desaparece en $\|u(t) - u_*(t)\|$.

donde $L = \frac{\partial}{\partial t} F(u_0, t_0)$ es el Jacobiano de F particularizado en la solución $u(t)$ cuya estabilidad deseamos estudiar y denotaremos por λ_k a los autovalores de L. La matriz L es diagonalizable cuando la multiplicidad algebraica y geométrica[4] de todos sus autovalores es la misma. Entonces podemos realizar un cambio de base $u = Qv$ con Q la matriz formada por los autovectores de L y la ecuación (3.43) toma la forma

$$\frac{\mathrm{d}v_{L_k}}{\mathrm{d}t} = \lambda_k v_{L_k}, \quad k = 1, ..., s. \tag{3.44}$$

La soluciones de (3.44) son de la forma $v_k = Ce^{\lambda_k t}$, con C una constante. Cuando la matriz L no es diagonalizable podemos utilizar la forma canónica de Jordan y las soluciones serán de la misma forma excepto para aquellos autovalores con multiplicidad algebraica diferente a su multiplicidad geométrica, en cuyo caso serán del tipo $v_k = Ct^m e^{\lambda_k t}$ con $m \geq 1$.
A diferencia de las ecuaciones no lineales, todas las soluciones de las ecuaciones lineales tienen el mismo carácter de estabilidad, es decir, podemos hablar de la *estabilidad de la ecuación lineal*. Para que el análisis de estabilidad lineal nos sea de utilidad necesitamos conocer la relación entre la estabilidad de la solución lineal $u_L(t)$ y la de la ecuación diferencial completa $u(t)$:

- Si $u_L(t)$ es asintóticamente estable $\Longrightarrow$ $u(t)$ es estable.
- Si $u_L(t)$ es inestable $\Longrightarrow$ $u(t)$ es inestable.
- Si $u_L(t)$ es estable $\Longrightarrow$ no se puede afirmar nada de $u(t)$.

Una vez hecha la conexión entre la estabilidad de $u_L(t)$ y $u(t)$ pasamos a estudiar la estabilidad del sistema lineal (3.44):

- Si todos los autovalores cumplen que $Re(\lambda_k) < 0 \Longrightarrow \|u_L(t)\| \to 0$ es asintóticamente estable. Re significa parte real.
- Si todo los autovalores cumplen $Re(\lambda_k) \leq 0$ y aquellos autovalores con $Re(\lambda_k) = 0$ tienen la misma multiplicidad algebraica y geométrica $\Longrightarrow$ $u_L(t)$ es estable.
- $u_L(t)$ es inestable en cualquier otro caso.

[4]La multiplicidad geométrica de un autovalor es la dimensión del espacio de sus autovectores asociados. La multiplicidad algebraica de un autovalor orden de dicho autovalor como cero del polinomio característico de L.

3.5.3 Consistencia, estabilidad y convergencia del esquema numérico

Una vez estudiada la existencia y unicidad del problema que deseamos resolver podemos pasar a analizar los diferentes esquemas numéricos. La mayor parte de los esquemas numéricos pueden expresarse de la forma

$$\sum_{j=0}^{p} \alpha_j u^{n+1-j} = \Delta t H(u^{n+1}, ..., u^{n+1-p}, t_n, ...) \tag{3.45}$$

donde p es el número de pasos, α_j constantes del esquema y H una función que depende de $F(u,t)$ y del esquema. La propiedad más importante que debe satisfacer un esquema numérico es la *convergencia*.
Un esquema numérico es convergente si es capaz de obtener la solución exacta del problema de condiciones iniciales cuando el paso temporal se hace infinitamente pequeño.

Definición 3 *Se dice que un método numérico es convergente si para todo problema de condiciones iniciales bien planteado cumple que*

$$\lim_{\Delta t \to 0} u^n = u(t_n), \tag{3.46}$$

para todas las soluciones numéricas u^n.

Evidentemente ésta es una propiedad deseada para el esquema numérico. Para comprobar si un esquema es convergente no se utiliza la relación (3.46) sino que se hace uso del *teorema de Lax*.

Teorema 4 *(Teorema de Lax). Para un problema de condiciones iniciales bien planteado, las condiciones necesarias y suficientes para que un esquema numérico sea convergente son que sea consistente y estable.*

Si un esquema numérico no es convergente se dice que es divergente. Podemos hacer la siguiente clasificación:

- Divergencia explosiva: la aproximación no converge a la solución para $\Delta t \to 0$ (esquema inestable).
- Divergencia a otra solución: para $\Delta t \to 0$ converge a otra solución diferente (esquema no consistente).
- Convergencia condicional: el esquema converge a la solución cuando $\Delta t \to 0$ y para valores de $\Delta t < \Delta t_{max}$ no diverge.

- Convergencia incondicional: el esquema converge a la solución cuando $\Delta t \to 0$ y nunca diverge independientemente de Δt.

Pasamos ahora a definir los conceptos de consistencia y estabilidad de un esquema numérico.

- **Consistencia**
  La consistencia indica la bondad con la que un esquema numérico representa la ecuación diferencial original cuando el paso temporal se hace infinitamente pequeño. Para definir la consistencia es útil utilizar el concepto de residuo definido como

$$R^{n+1} = \sum_{j=0}^{p} \alpha_j u(t_{n+1-j}) - \Delta t H(u(t_{n+1}), ..., u(t_{n+1-p}), t_n, ...), \quad (3.47)$$

  que consiste en tomar la solución exacta del problema $u(t)$ e introducirla en el esquema numérico. Esta definición es idéntica a la de error de truncación introducida en §3.4.

  Definición 4 *Se dice que un esquema numérico es consistente si para todo problema de condiciones iniciales bien planteado el residuo* R^{n+1} *cumple*

$$\lim_{\Delta t \to 0} \frac{R^{n+1}}{\Delta t} = 0, \quad (3.48)$$

  Las condiciones necesarias y suficientes para que un esquema numérico sea consistente son

$$\sum_{j=0}^{p} \quad \alpha_j = 0, \quad (3.49)$$

$$\sum_{j=0}^{p} \quad j\alpha_j + \frac{H(u(t_{n+1}), ..., u(t_{n+1}), t_{n+1}, ...)}{F(u(t_{n+1}), t_{n+1})} = 0, \quad (3.50)$$

  en el límite $\Delta t \to 0$. Un esquema consistente tiene un error de truncación al menos de $o(\Delta t^2)$. En el caso de los esquemas Runge-Kutta las condiciones para la consistencia son

$$\sum_{i=1}^{e} b_i = 1, \quad (3.51)$$

  además en general supondremos que

$$\sum_{j=1}^{e} a_{ij} = c_i. \quad (3.52)$$

- **Estabilidad del esquema numérico**
  En general, queremos que el carácter de estabilidad del esquema numérico aplicado a una problema estable de condiciones iniciales sea el mismo que el de dicho problema. El parámetro libre en un esquema numérico es Δt y buscaremos cuál es el Δt_{max} para el cual el esquema numérico es estable cuando $\Delta t < \Delta t_{max}$. La estabilidad no lineal depende tanto del esquema numérico como de la ecuación diferencial y sus condiciones iniciales. Al igual que ocurría en el problema de condiciones iniciales, estudiar la estabilidad no lineal puede ser una tarea muy complicada y en su lugar suele estudiarse la estabilidad del problema de condiciones iniciales lineal de la forma

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u, \tag{3.53}$$

  con λ el autovalor del problema con parte real e imaginaria $\lambda = \lambda_r + i\lambda_i$. La estabilidad lineal del esquema numérico se obtiene estudiando la ecuación en diferencias que resulta de aplicar el esquema numérico (3.45) al problema (3.53). Al igual que la ecuación diferencial lineal admite soluciones del tipo $e^{\lambda t}$, la ecuación en diferencias admite aquellas de la forma r^n (ver nota[5]). Introduciendo $u^n = r^n$ en la ecuación (3.45) aplicada al problema (3.53) obtenemos el denominado *polinomio de estabilidad del esquema numérico* que será de la forma

$$\Pi(r) = \sum_{j=0}^{p} (\alpha_j - \Delta t \lambda f_j(\Delta t \lambda)) r^{p-j} = 0, \tag{3.54}$$

  donde las funciones f_j dependerán del esquema numérico. Dado que estamos buscando soluciones del tipo $u^n = r^n$, el carácter de estabilidad dependerá del valor de r que a su vez será función de $\Delta t \lambda$.

  Teorema 5 *Un esquema numérico es absolutamente estable para un Δt dado si todas las raices del polinomio de estabilidad satisfacen $|r_k| < 1$, $k = 1, ..., p$, para todo autovalor dado del problema (3.53).*

  La solución de la ecuación en diferencias también puede ser expresada como $u^n = u^0 \sigma^n$, donde σ es el *factor de amplificación* que debe ser menor que uno para la estabilidad de la solución.

[5]Es importante notar que r^n representa el número r elevado a la n-ésima potencia, mientras que por notación hemos adoptado $u^n = u(t_n)$ y $F^n = F(u(t_n), t_n)$ que significa u y F evaluadas en el instante t_n y no su potencia.

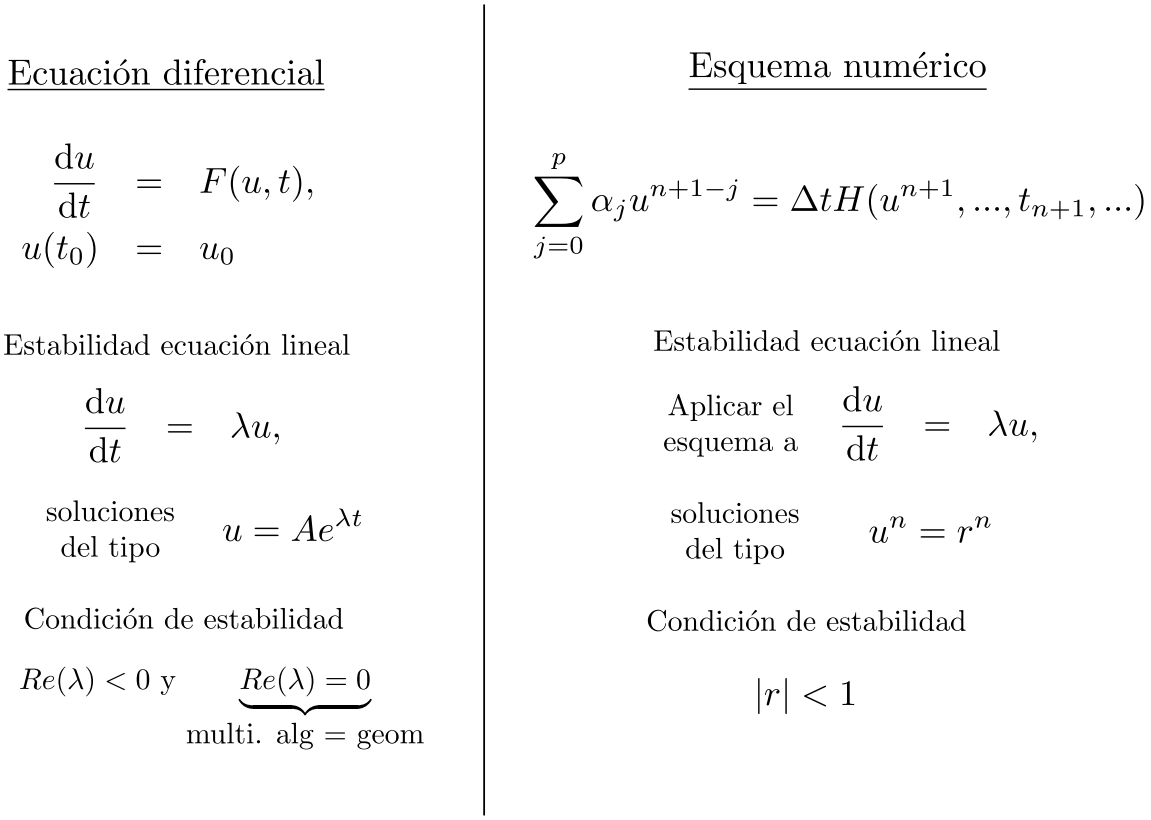


Figura 3.5: Tabla resumen del estudio de estabilidad lineal en ecuaciones diferenciales ordinarias y esquemas numéricos.

- **Región de estabilidad absoluta**
  Para visualizar de forma más clara el valor apropiado de Δt en función de los valores de λ, haremos uso de la *región de estabilidad*, definida por la región $|r| \leq 1$ en unos ejes ($\Delta t\lambda_r$,$\Delta t\lambda_i$). La región de estabilidad nos proporciona la relación entre la estabilidad de la ecuación diferencial lineal ($\lambda_r \leq 0$) y el esquema numérico ($|r| < 1$). Un método convergente incluirá $\Delta t = 0$ en la región de estabilidad. Definimos el número complejo ω como

$$\omega = \Delta t\lambda = \Delta t(\lambda_r + i\lambda_i), \tag{3.55}$$

  y el polinomio característico de estabilidad queda

$$\Pi(r) = \sum_{j=0}^{p} (\alpha_j - \omega f_j(\omega)) r^{p-j} = 0. \tag{3.56}$$

  Sus raíces son números complejos que podemos expresar como $r = r_0 e^{i\theta}$. La región de estabilidad absoluta está definida por aquellas zonas con $r_0 = 1$ y su frontera por

$$\sum_{j=0}^{p} (\alpha_j - \omega f_j(\omega)) \left(e^{i\theta}\right)^{p-j} = 0, \tag{3.57}$$

  que nos proporciona de forma implícita la ecuación de la frontera $\omega = \omega(\theta)$. En muchas ocasiones no se puede obtener analíticamente la función de $\omega = \omega(\theta)$ y se tiene que resolver numéricamente.

- **Soluciones espúreas**
  Las soluciones espúreas son soluciones falsas producidas por el esquema numérico. Están ligadas al orden de la ecuación en diferencias. Cuando buscamos soluciones del tipo $u^n = r^n$, una ecuación en diferencias de orden p dará lugar a p raices r, aunque la ecuación diferencial que aproxima tiene solución única. En general, los esquemas multipaso de p pasos tienen $p - 1$ raíces espúreas que hay que controlar y evitar que emerjan. Los esquemas unipaso (y multietapa) no presentan este problema.

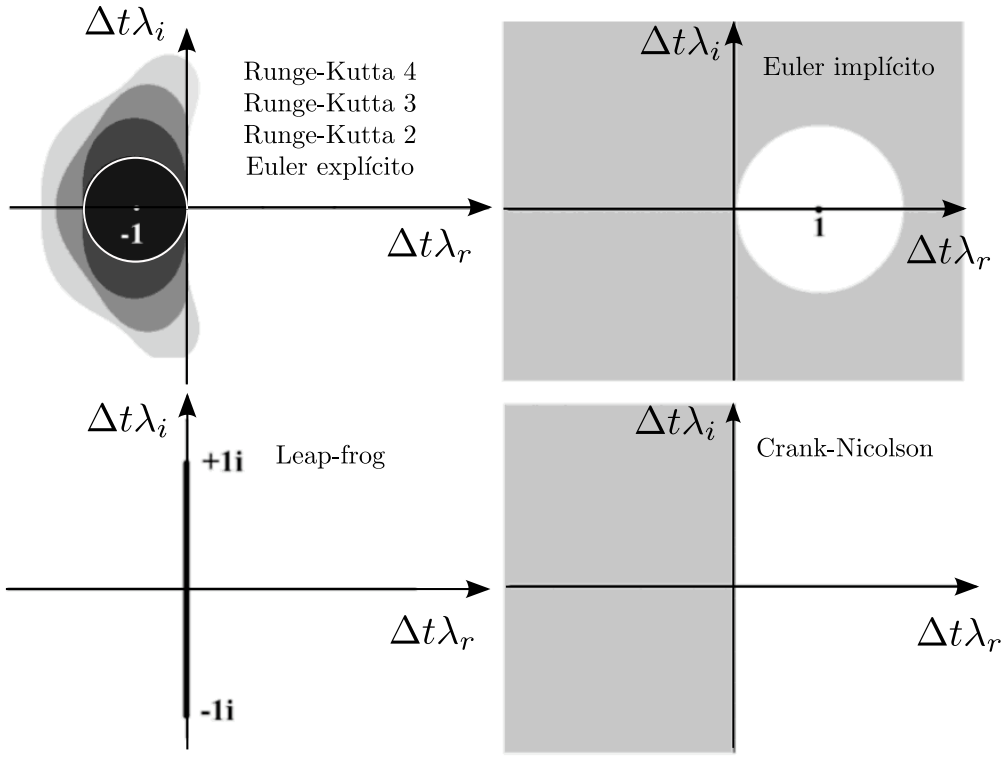


Figura 3.6: Regiones de estabilidad para diferentes esquemas numéricos. La zonas oscuras representan la región de estabilidad.

Capítulo 4

Estrategias de resolución de las ecuaciones de la Mecánica de Fluidos

Se pueden utilizar diferentes estrategias para la resolución numérica de las ecuaciones de Navier-Stokes dependiendo de qué formulación se escoja así como de los pasos a seguir para resolverla independientemente de la discretización espacial y temporal que se utilice. A lo largo del capítulo, utilizaremos las variables en negrita para representar vectores. Por cuestiones de extensión, este capítulo está restringido al caso de flujos incompresibles. Consideraremos flujos con viscosidad y densidad constantes y sometidos a fuerzas irrotacionales.

4.1 Formulación con presión

La mayor dificultad al obtener soluciones precisas para flujos incompresibles reside en que la ecuación de continuidad no tiene explícitamente término con derivada temporal,

$$\nabla \cdot \boldsymbol{v} = 0. \tag{4.1}$$

Es decir, no existe una ecuación de evolución para $\partial p/\partial t = rhs(t)$. La restricción de conservar la masa se consigue mediante el acoplamiento implícito de la ecuación de continuidad con la de cantidad de movimiento a través de la presión.

4.1.1 Método de proyección

Los métodos de proyección de velocidad (también llamados en la literatura de corrección de presión o de paso fraccionado) se basan en obtener primero una solución de la velocidad que no cumple la ecuación de continuidad para luego corregirla (proyectarla) forzando a que sea solenoidal. Existen múltiples variantes del método y aquí nos limitamos a usar una de tantas. Partimos de las ecuaciones de Navier-Stokes para flujos incompresibles adimensionalizada,

$$\begin{aligned} \frac{\partial \boldsymbol{v}}{\partial t} &= -\boldsymbol{v}\cdot\nabla\boldsymbol{v} - \nabla p + \frac{1}{Re}\Delta\boldsymbol{v} + \boldsymbol{f}, && (4.2)\\ \nabla\cdot\boldsymbol{v} &= 0. && (4.3) \end{aligned}$$

En el desarrollo siguiente, por simplicidad, utilizamos un esquema Euler explícito para la discretización temporal. Sin embargo el proceso a seguir es similar cuando se parte de otra discretización. La discretización espacial es genérica. Denotaremos por ∇_δ a la aproximación numérica de las derivadas espaciales y por Δ_δ a la aproximación numérica del Laplaciano. Los superíndices n y $n+1$ se refieren a las soluciones en los instantes t_n y t_{n+1}. El resultado de discretizar las ecuaciones (4.2) es

$$\begin{aligned} \boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \Delta t\left(-\boldsymbol{v}^n\cdot\nabla_\delta\boldsymbol{v}^n - \nabla_\delta p^{n+1} + \frac{1}{Re}\Delta_\delta\boldsymbol{v}^n + \boldsymbol{f}^n\right), && (4.4)\\ \nabla_\delta\cdot\boldsymbol{v}^{n+1} &= 0. && (4.5) \end{aligned}$$

El problema de la ecuación anterior es que desconocemos el término $\nabla_\delta p^{n+1}$. Los métodos de correción de presión eliminan este término junto con la ecuación de continuidad $\nabla_\delta\cdot\boldsymbol{v}^{n+1}=0$, lo que resulta

$$\boldsymbol{v}^* = \boldsymbol{v}^n + \Delta t\left(-\boldsymbol{v}^n\cdot\nabla_\delta\boldsymbol{v}^n + \frac{1}{Re}\Delta_\delta\boldsymbol{v}^n + \boldsymbol{f}^n\right). \quad (4.6)$$

Como consecuencia, ya no obtenemos $\boldsymbol{v}^{n+1}$ que satisface $\nabla_\delta\cdot\boldsymbol{v}^{n+1}=0$ sino otra solución $\boldsymbol{v}^*$ que en general cumple que $\nabla_\delta\cdot\boldsymbol{v}^*\neq 0$. La diferencia entre las ecuaciones (4.4) y (4.6) es

$$\boldsymbol{v}^{n+1} - \boldsymbol{v}^* = -\Delta t\nabla_\delta p^{n+1}, \quad (4.7)$$

es decir, podemos recuperar la ecuación (4.4) sumando (4.7) y (4.6). Si tomamos la divergencia numérica de (4.7) resulta,

$$\nabla_\delta\cdot\boldsymbol{v}^{n+1} - \nabla_\delta\cdot\boldsymbol{v}^* = -\Delta t\Delta_\delta p^{n+1}, \quad (4.8)$$

sin embargo, estamos buscando $\nabla_\delta \cdot \boldsymbol{v}^{n+1} = 0$ por lo que

$$\Delta_\delta p^{n+1} = \frac{\nabla_\delta \cdot \boldsymbol{v}^*}{\Delta t}. \tag{4.9}$$

La ecuación anterior es una ecuación de Poisson que nos indica cuánto tiene que valer la presión para corregir $\boldsymbol{v}^*$ y obtener $\boldsymbol{v}^{n+1}$. Es importante notar que (4.9) no es la ecuacion de la presión real del fluido, sino más bien una pseudo-presión que garantiza que la divergencia del campo final sea nula. Esta ecuación es la que vamos a utilizar en lugar de $\nabla_\delta \cdot \boldsymbol{v}^{n+1} = 0$. Necesitamos una condición de contorno para (4.9) y aunque no entramos en detalle, se puede demostrar que debe ser de tipo Neumann $\partial p/\partial n = 0$ donde n es la dirección normal a la frontera. A lo largo de todo el desarrollo anterior, hemos utilizado p^{n+1} y no p^n. La razón radica en que la presión es una variable instantánea, y dado que queremos conseguir $\nabla_\delta \cdot \boldsymbol{v}^{n+1} = 0$ necesitamos usar p en el instante t_{n+1}. Por ello, el término de la presión debe discretizarse siempre con un esquema temporal implícito. Por defecto, se suele utilizar Euler implícito tal y como se ha hecho en el desarrollo anterior y con la notación p en lugar de p^{n+1}.
Con toda la información anterior, estamos en condiciones de resolver numéricamente las ecuaciones de Navier-Stokes. El proceso es el siguiente:

1. Se parte de una solución $\boldsymbol{v}^n$ en el instante t_n que cumpla $\nabla_\delta \cdot \boldsymbol{v}^n = 0$.

2. Se obtiene la aproximación con divergencia no nula $\boldsymbol{v}^*$:

$$\boldsymbol{v}^* = \boldsymbol{v}^n + \Delta t \left(-\boldsymbol{v}^n \cdot \nabla_\delta \boldsymbol{v}^n + \frac{1}{Re} \Delta_\delta \boldsymbol{v}^n + \boldsymbol{f}^n \right). \tag{4.10}$$

3. Se obtiene la presión necesaria p^{n+1} para corregir v^* con la ecuación de Poisson:

$$\Delta_\delta p^{n+1} = \frac{\nabla_\delta \cdot \boldsymbol{v}^*}{\Delta t}. \tag{4.11}$$

4. Se halla el gradiente de presiones $\nabla_\delta p^{n+1}$.

5. Se corrige la velocidad v^* para que tenga divergencia nula (proyección de v^*):

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^* - \Delta t \nabla_\delta p^{n+1}. \tag{4.12}$$

6. Se repite el proceso desde el paso 1.

El proceso es idéntico en el caso de utilizar otra discretización temporal en el paso 2 de la forma

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t H(\boldsymbol{v}^{n+1}, \boldsymbol{v}^n, ...), \tag{4.13}$$

simplemente sustituimos $\boldsymbol{v}^{n+1}$ por $\boldsymbol{v}^*$, eliminamos los términos de presión que aparezcan en la función H y resolvemos el sistema implícito de ecuaciones para obtener $\boldsymbol{v}^*$. La ecuación de Poisson para la pressión y la ecuación de corrección para sacar $\boldsymbol{v}^{n+1}$ se obtienen siguiendo los pasos descritos anteriormente. Generalmente los términos convectivos se discretizan con un esquema temporal explícito (ver sección §5.3), sin embargo, en algunas ocasiones interesa usar esquemas implícitos para poder avanzar más rápidamente en el tiempo, por ejemplo, para alcanzar el estado estacionario. En estos casos aparecen dificultades dado que no hay forma de eliminar el término convectivo $\boldsymbol{v}^{n+1} \cdot \nabla \boldsymbol{v}^{n+1}$, incluso si tomamos la divergencia. El problema se resuelve de forma iterativa con métodos denominados SIMPLE (Semi-Implicit Method for Pressure Linked Equations) del cual existen múltiples variantes (SIMPLER, SIMPLEC, SIMPLEST, PISO). Es importante remarcar que ésta es una de tantas formas de atacar el problema y en la literatura se pueden encontrar variantes de todo tipo.

4.2 Formulación sin presión

En la sección hemos visto que la mayor parte de los problemas a la hora de resolver las ecuaciones en formulación con variables primitivas (velocidades y presión) vienen del término de presión. El término de presión, sin embargo, no es un campo cualquiera sino que proviene de un gradiente y es por lo tanto irrotacional y cumple,

$$\nabla \times \nabla p = 0. \tag{4.14}$$

Esta característica se puede utilizar para eliminar ∇p de las ecuaciones de cantidad de movimiento y obtener una formulación sin presión. Si definimos la vorticidad como

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{v}, \tag{4.15}$$

y tomamos el rotor de la ecuación de cantidad de movimiento adimensionalizada,

$$\nabla \times \left[\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{v} - \boldsymbol{f} \right] = 0 \tag{4.16}$$

el resultado es la ecuación de evolución de la vorticidad,

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{\omega} = \boldsymbol{\omega} \cdot \nabla \boldsymbol{v} + \frac{1}{Re} \Delta \boldsymbol{\omega} + \nabla \times \boldsymbol{f}, \tag{4.17}$$

en la cual no hay término de presión. Las estrategias de resolución con formulación sin presión se basan de una forma u otra en la ecuación (4.17). Uno de los inconvenientes de ésta formulación reside en la imposición de las condiciones de contorno pues generalmente se especifican las velocidades en la frontera del dominio pero no las vorticidades.

4.2.1 Vorticidad-función de corriente

Para flujos incompresibles en 2D con propiedades fluidas constantes, las ecuaciones de Navier-Stokes pueden ser simplificadas utilizando la *función de corriente* ψ y la vorticidad ω como variables dependientes. La función de corriente satisface

$$\frac{\partial \psi}{\partial y} = u, \quad \frac{\partial \psi}{\partial x} = -v, \tag{4.18}$$

donde u y v son las velocidades en las direcciones x e y respectivamente y garantiza que el flujo sea incompresible pues,

$$\nabla \cdot \boldsymbol{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = \frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial x \partial y} = 0. \tag{4.19}$$

Las líneas $\psi = constante$ son líneas de corriente (aquellas que son tangentes en todo punto al vector velocidad). En dos dimensiones, el vector vorticidad tiene únicamente una componente no nula

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \tag{4.20}$$

que es ortogonal al plano donde se encuentra contenido el movimiento del fluido. Combinando las ecuaciones (4.18) y (4.20) obtenemos la relación que tiene que satisfacer ψ,

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega, \tag{4.21}$$

que se trata de la ecuación de Poisson cuyo término fuente es la vorticidad. Por último, la ecuación de la vorticidad en dos dimensionas se obtiene a partir de la expresión (4.17),

$$\frac{\partial \omega}{\partial t} + u \frac{\partial \omega}{\partial x} + v \frac{\partial \omega}{\partial y} = \frac{1}{Re} \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right). \tag{4.22}$$

Con ésto tenemos la formulación completada. Las incógnitas son ω, ψ, u y v que están determinadas por las ecuaciones (4.18), (4.21) y (4.22) junto con las condiciones iniciales y de contorno apropiadas. En el caso de que sea necesaria la presión, se puede obtener resolviendo la ecuación de la presión que se obtiene tomando la divergencia de la ecuación de cantidad de movimiento,

$$\Delta p = -\nabla \cdot (\boldsymbol{v} \cdot \nabla \boldsymbol{v}) + g, \tag{4.23}$$

donde g contiene lo términos derivados de fuerzas de volumen sobre el fluido. Denotaremos por δ la aproximación numérica de las derivadas espaciales y utilizaremos los superíndices n y $n+1$ para referirnos a las soluciones en los instantes t_n y t_{n+1}. El proceso de resolución es el siguiente:

1. Se parte de un campo de velocidades inicial o procedente del paso anterior u^n y v^n.

2. Se calcula la vorticidad con

$$\omega^n = \frac{\delta v^n}{\delta x} - \frac{\delta u^n}{\delta y}. \tag{4.24}$$

3. Se avanza ω^n un paso en el tiempo usando

$$\omega^{n+1} = \omega^n + \Delta t H(\omega^{n+1}, \omega^n, ...) \tag{4.25}$$

   donde H se obtiene particularizando en esquema de avance temporal para la ecuación

$$\frac{\partial \omega}{\partial t} = -u\frac{\partial \omega}{\partial x} - v\frac{\partial \omega}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right). \tag{4.26}$$

4. Se obtiene la función de corriente en el siguiente instante de tiempo

$$\frac{\delta^2 \psi^{n+1}}{\delta x^2} + \frac{\delta^2 \psi^{n+1}}{\delta y^2} = -\omega^{n+1}, \tag{4.27}$$

5. Se calculan las nuevas velocidades

$$\frac{\delta \psi^{n+1}}{\delta y} = u^{n+1}, \quad \frac{\delta \psi^{n+1}}{\delta x} = -v^{n+1}, \tag{4.28}$$

6. Se repite el proceso desde el paso 1.

El proceso anterior es sencillo siempre y cuando el esquema numérico elegido para el paso 3 no contenga velocidades en el paso $n+1$, lo cual se consigue haciendo explícito el término de convección.

4.2.2 Vorticidad-velocidad

En el caso tridimensional la función de corriente es más complicada. Por ello, se puede utilizar una formulación basada en una componente de velocidad y la componente de vorticidad en la misma dirección, por el ejemplo v y ω_y. El objetivo es siempre eliminar la presión.
(En proceso...)

Capítulo 5

Discretización espacial: diferencias finitas

El primer paso para obtener un esquema numérico espacial es discretizar la geometría del dominio donde se quieren resolver las ecuaciones de Navier-Stokes, es decir, definimos una *malla*. Las diferencias finitas son generalmente utilizadas en mallas estructuradas, aunque no tienen que ser necesariamente uniformes. Las intersecciones entre las líneas de la malla se denominan nodos. El objetivo de la discretización espacial por diferencias finitas es sustituir las ecuaciones en derivadas parciales por ecuaciones en diferencias cuyas incógnitas son las velocidades (u otras magnitudes físicas) en los nodos interiores de la malla. Los valores en el contorno son conocidos cuando se imponen directamente como condición de contorno (condición tipo Dirichlet) y desconocidos cuando se imponen las derivadas (condición tipo Neumann). No todas las variables tienen que estar definidas en la misma malla, por ejemplo, la velocidad u puede estar definidas en una malla y la velocidad v en otra. A este tipo de mallas se las denomina *stagger* y veremos su utilidad más adelante. En los casos más sencillos y en 2D, las mallas tendrán una forma como la que muestra la figura 5.1.
Sin pérdida de generalidad, utilizaremos la ecuación lineal unidimensional de convección-difusión o también llamada ecuación de Burgers viscosa,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{5.1}$$

donde $u = u(x,t)$, c es la velocidad de convección y ν la viscosidad. ¿Cómo pasamos de $u(x,t)$ a su aproximación numérica discreta? La idea básica de las diferencias finitas proviene de la definición de derivada

$$\frac{\partial u}{\partial x}(x_i) = \lim_{\Delta x \to 0} \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x}. \tag{5.2}$$

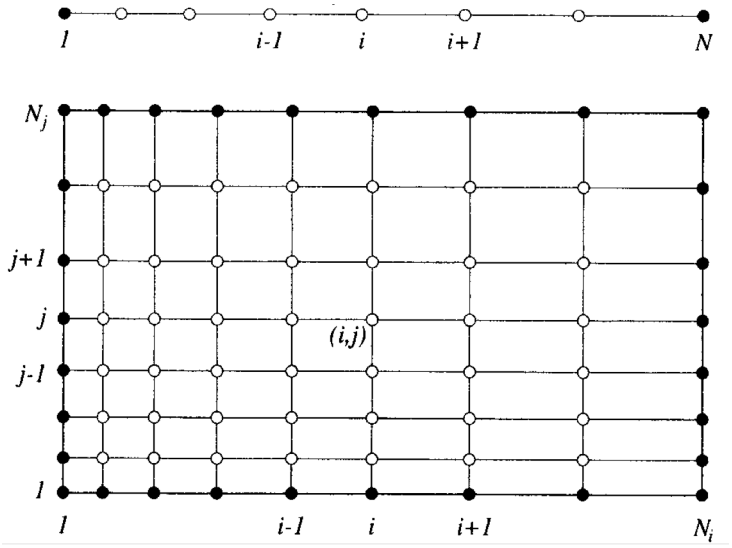


Figura 5.1: Ejemplo de malla de colocación 1D y 2D. Los nodos está representados por círculos. Aquellos que están en la frontera aparecen como círculos cerrados.

Podemos utilizar la definición anterior para aproximar la derivada tomando $\Delta x > 0$ pero pequeño. Como consecuencia, aparecerá un error asociado.
A lo largo del capítulo utilizaremos la siguiente notación:

- $u(x,t)$: solución exacta de la ecuación (5.1).
- u_i^n: aproximación numérica de $u(x,t)$ en el instante t_n y el punto x_i. En el caso de problemas 2D, usaremos $u_{i,j}^n$ donde ahora el punto espacial es (x_i, y_j).
- Δt: paso de tiempo entre dos soluciones numéricas. Puede variar en cada instante.
- Δx_i: distancia entre dos nodos consecutivos $x_{i+1} - x_i$. Para mallas uniformes usaremos directamente Δx.
- $\left(\frac{\partial^n u}{\partial x^n}\right)_i$: valor exacto de la derivada n-ésima en x_i.
- $\left(\widetilde{\frac{\partial^n u}{\partial x^n}}\right)_i$: valor aproximado de la derivada n-ésima en x_i.

5.1 Obtención de esquemas

En la literatura se pueden encontrar varios métodos para obtener esquemas de diferencias finitas. Los más clásicos son por *desarrollo en serie de Taylor* y por *ajuste polinómico*.

- **Desarrollo en serie de Taylor**:

  Toda función continua y diferenciable $u(x)$ puede, en la vecindad de x_i, ser expresada como una serie de Taylor de la forma

$$u(x) = u(x_i) + (x - x_i)\left(\frac{\partial u}{\partial x}\right)_i + \frac{(x - x_i)^2}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_i \tag{5.3}$$

$$+... + \frac{(x - x_i)^n}{n!}\left(\frac{\partial^n u}{\partial x^n}\right)_i + N_i. \tag{5.4}$$

  Particularizando la expresión anterior en diferentes puntos $x = x_j$, permite obtener un sistema de ecuaciones para despejar las derivadas $\left(\frac{\partial^n u}{\partial x^n}\right)_i$ en función del valor de u_j y x_j en varios nodos. Por ejemplo, si desarrollamos la serie (5.3) hasta $n = 3$, aparecen tres incógnitas, $\left(\frac{\partial^n u}{\partial x^n}\right)_i$, $n = 1, .., 3$, por lo que particularizamos la serie en tres puntos, x_{i+2}, x_{i+1} y x_{i-1} despreciando los términos, N_i, los cuales determinarán el error del esquema de diferencias finitas. Hay que tener en cuenta que los términos N_i serán pequeños frente al resto siempre que tomemos Δx pequeño.

- **Ajuste polinómico**:

  Aproximamos u con una función de interpolación que pasa por los puntos x_i

$$u(x) \approx \sum L_i(x) u_i, \tag{5.5}$$

  donde las funciones $L_i(x)$ dependen de la interpolación utilizada (por ejemplo, de Lagrange). Hallamos su derivada y la particularizamos en $x = x_i$

$$\left(\frac{\partial^n u}{\partial x^n}\right)_i \approx \sum \left(\frac{\partial^n L_i}{\partial x^n}\right)_i u_i, \tag{5.6}$$

  con lo que ya tenemos el esquema de diferencias finitas.

Todos los esquema se pueden deducir tanto por un método como por el otro. Si denotamos *diferencias finitas* por DF, los más sencillos son:

- Aproximación de la derivada primera:
  - DF atrasadas:

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i \approx (u_i - u_{i-1})/\Delta x + \frac{\Delta x}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i. \tag{5.7}$$

  - DF adelantadas:

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i \approx (u_{i+1} - u_i)/\Delta x - \frac{\Delta x}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i. \tag{5.8}$$

  - DF centradas:

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i \approx (u_{i+1} - u_{i-1})/2\Delta x - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i. \tag{5.9}$$

- Aproximación de la derivada segunda:
  - DF atrasadas:

$$\left(\widetilde{\frac{\partial^2 u}{\partial x^2}}\right)_i \approx \frac{u_i - 2u_{i-1} + u_{i-2}}{\Delta x^2} + \Delta x\left(\frac{\partial^3 u}{\partial x^3}\right)_i. \tag{5.10}$$

  - DF adelantadas:

$$\left(\widetilde{\frac{\partial^2 u}{\partial x^2}}\right)_i \approx \frac{u_{i+2} - 2u_{i+1} + u_i}{\Delta x^2} - \Delta x\left(\frac{\partial^3 u}{\partial x^3}\right)_i. \tag{5.11}$$

  - DF centradas:

$$\left(\widetilde{\frac{\partial^2 u}{\partial x^2}}\right)_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - \frac{(\Delta x)^2}{12}\left(\frac{\partial^4 u}{\partial x^4}\right)_i. \tag{5.12}$$

Los términos en azul se desprecian a la hora de usar cada esquema. Representan el error dominante y los trataremos en la siguiente sección. A los esquemas que utilizan información de puntos situados únicamente a la derecha o izquierda del nodo x_i se les denominan esquemas *upwind* (DF atrasadas y adelantadas).

5.2 Análisis de errores

Cualquiera de los métodos anteriores da lugar a un error en la aproximación de las derivadas, bien sea porque truncamos la serie de Taylor o bien porque usamos un número finito de nodos en la función de interpolación. Los errores están relacionados con los términos despreciados N_i. En general, podemos definir el error del esquema como

$$T_i = \left(\frac{\partial^n u}{\partial x^n}\right)_i - \left(\widetilde{\frac{\partial^n u}{\partial x^n}}\right)_i , \tag{5.13}$$

donde $\widetilde{\frac{\partial^n u}{\partial x^n}}$ es el valor de la derivada aproximado por el esquema de diferencias finitas. Los errores cambian en cada punto x_i y dependen tanto de la forma que tenga la función u como de la malla. Desde el punto de vista matemático, el error es simplemente aquella cantidad que hay que añadir al esquema para obtener exactamente el valor de $(\partial^n u/\partial x^n)$. Sin embargo, este error se puede analizar e interpretar desde diferentes puntos de vista, y ésto es lo que hacemos a continuación.

5.2.1 Error de truncación

Podemos entender el error como consecuencia directa de truncar la serie de Taylor. Sin embargo, no siempre nos interesa el valor absoluto del error cometido (que depende de u, de la forma de la malla y del punto de la malla) sino que queremos saber cómo de rápido mejora la aproximación de la derivada al refinar la malla, es decir, al hacer $\Delta x \to 0$. Dado que los errores para mallas uniformes son de la forma,

$$T_i = \sum_{l=m}^{\infty} A_l \frac{(\Delta x)^{l-n}}{l!} \left(\frac{\partial^l u}{\partial x^l}\right)_i , \tag{5.14}$$

donde A_l son constantes que dependen del esquema y n se corresponde con la derivada aproximada $\partial^n u/\partial x^n$. El término más importante cuando $\Delta x \to 0$ será

$$T_i \approx A_m \frac{(\Delta x)^{m-n}}{m!} \left(\frac{\partial^m u}{\partial x^m}\right)_i . \tag{5.15}$$

Se denomina orden del esquema al exponente $m-n$. El error T_i se puede hallar fácilmente reteniendo los términos de orden superior al obtener el esquema por desarrollo en serie de Taylor. El error para mallas no uniformes se obtiene de forma análoga usando $\Delta x = x_j - x_i$ según corresponda. Un mismo esquema aplicado a mallas uniformes y no uniformes puede tener diferente orden.

5.2.2 Error de disipación y dispersión

Los errores de la solución numérica pueden ser interpretados como términos con significado físico que añadimos a la ecuación. Estos términos son despreciables para Δx suficientemente pequeño, sin embargo, es necesario entender cuál es su efecto en la solución numérica. Para ello, usaremos el **análisis de ecuación modificada**. Consideremos la ecuación,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{5.16}$$

y aproximemos las derivadas espaciales en una malla uniforme por

$$\frac{\partial u}{\partial x} \approx \widetilde{\frac{\partial u}{\partial x}} + A\frac{(\Delta x)^{m-1}}{m!}\left(\frac{\partial^m u}{\partial x^m}\right), \tag{5.17}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \widetilde{\frac{\partial^2 u}{\partial x^2}} + B\frac{(\Delta x)^{r-2}}{r!}\left(\frac{\partial^r u}{\partial x^r}\right), \tag{5.18}$$

donde A y B son constantes y m y r dependen del orden del esquema. Para obtener la ecuación discretizada, los términos de mayor orden son despreciados y obtenemos,

$$\frac{\partial u}{\partial t} + c\widetilde{\frac{\partial u}{\partial x}} = \nu\widetilde{\frac{\partial^2 u}{\partial x^2}}. \tag{5.19}$$

Si tenemos en cuenta los términos despreciados en la ecuación (5.19) resulta,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} - cA\frac{(\Delta x)^{m-1}}{m!}\left(\frac{\partial^m u}{\partial x^m}\right) = \nu\frac{\partial^2 u}{\partial x^2} - \nu B\frac{(\Delta x)^{r-2}}{r!}\left(\frac{\partial^r u}{\partial x^r}\right), \tag{5.20}$$

que es la ecuación numérica que vamos a resolver a falta de discretizar en el tiempo. Cuando resolvemos numéricamente la ecuación (5.19) no estamos obteniendo una solución para el problema (5.16) sino la ecuación modificada (5.20) que tiene propiedades físicas diferentes. En general,

- Para m ó r pares aparecen términos (predominantemente) disipativos.
- Para m ó r impares aparecen términos (predominantemente) dispersivos.

Hay que tener en cuenta que estos términos son del orden $(\Delta x)^{m-1}$ y $(\Delta x)^{r-2}$, por lo que su efecto disminuye al refinar la malla y cuanto mayor sea el orden del esquema utilizado.
Como ejemplo, consideremos la siguiente ecuación discretizada espacialmente con diferencias finitas centradas tanto para la primera como para la segunda derivada,

$$\frac{\partial u}{\partial t} + c\frac{u_{i+1} - u_{i-1}}{2\Delta x} = \nu\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}. \tag{5.21}$$

Si $\Delta x \to 0$ entonces no hay errores y estamos resolviendo exactamente la ecuación (5.16). Sin embargo, desde el punto de vista numérico $\Delta x > 0$, por lo que aparecen errores. Para realizar el análisis de ecuación modificada intentamos recuperar la ecuación (5.16). Las aproximaciones usadas para las derivadas son

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right), \tag{5.22}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - \frac{(\Delta x)^2}{12}\left(\frac{\partial^4 u}{\partial x^4}\right). \tag{5.23}$$

Introduciendo las relaciones anteriores en (5.21) obtenemos la ecuación que estamos realmente resolviendo,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + c\frac{(\Delta x)^2}{6}\frac{\partial^3 u}{\partial x^3} = \nu\frac{\partial^2 u}{\partial x^2} + \nu\frac{(\Delta x)^2}{12}\frac{\partial^4 u}{\partial x^4}, \tag{5.24}$$

lo cual añade un término dispersivo,

$$c\frac{(\Delta x)^2}{6}\frac{\partial^3 u}{\partial x^3}, \tag{5.25}$$

y otro disipativo,

$$\nu\frac{(\Delta x)^2}{12}\frac{\partial^4 u}{\partial x^4}, \tag{5.26}$$

que serán muy pequeños si Δx es pequeño. En general, no hace falta realizar todo el proceso y obtener la ecuación modificada sino que podemos saber los errores mirando directamente las aproximaciones (5.22) y (5.23),

- Si el error tiene derivadas **pares** el esquema es (principalmente) **disipativo**.
- Si el error tiene derivadas **impares** el esquema es (principalmente) **dispersivo**.

5.2.3 Análisis de onda modificada

Si consideramos una función $u = \sin(kx)$, donde k es el número de onda y suponemos una malla uniforme con Δx, es intuitivo que cuanto mayor sea k , mayor será la frecuencia de u y llegará un momento en el que las oscilaciones de u sean igual o menores que Δx y no podremos estimar correctamente el valor de u ni de su derivada. Siguiendo esta idea, el análisis de onda modificada nos permite estudiar el error cometido por el esquema para diferentes

escalas (senos y cosenos de diferentes frecuencias). Suponemos un problema periódico con la función onda $u = e^{Ikx}$, donde I es la unida imaginaria, del cual sabemos que en un punto x_i se cumple que

$$\left(\frac{\partial u}{\partial x}\right)_i = Ike^{Ikx_i}. \tag{5.27}$$

Si aplicamos el esquema de diferencias finitas en una malla uniforme a $u = e^{Ikx}$ tendremos

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i = Ik_{ef}e^{Ikx_i}, \tag{5.28}$$

donde $k_{ef} = k_{ef}(k, \Delta x)$ es el **número de onda modificado** que depende del esquema utilizado y en general no es igual al valor teórico $k_{ef} = k$. Dado un tamaño de la malla Δx, el número de onda modificado nos indica cómo de bien estamos resolviendo la derivada del seno ó coseno con número de onda k. En general, k_{ef} es un número complejo y hay que comparar su parte real con k y su parte imaginaria con 0. Las desviaciones de la parte real de k_{ef} con respecto a k implica errores en la amplitud de la derivada (atenuación o amplificación) mientras que la parte imaginaria de k_{ef} está relacionada con errores en la fase. Por ejemplo, para el siguiente esquema de diferencias finitas centradas,

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \tag{5.29}$$

al aplicarlo a $u = e^{ikx}$ obtenemos,

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i = \frac{e^{Ikx_{i+1}} - e^{Ikx_{i-1}}}{2\Delta x}, \tag{5.30}$$

y dado que $x_{i+1} = x_i + \Delta x$ y $x_{i-1} = x_i - \Delta x$ resulta,

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i = I\frac{\sin(k\Delta x)}{\Delta x}e^{Ikx_i} = Ik_{ef}e^{Ikx_i}. \tag{5.31}$$

Se suele representar la función de onda modificada normalizada $k^*_{ef} = k_{ef}(k^*)/k_{max}$, donde $k^* = k/k_{max}$ y $k_{max} = \pi/\Delta x$, de tal forma que toda desviación de la recta $k^*_{ef} = k^*$ implica un error tal y como se muestra en la figura 5.2.

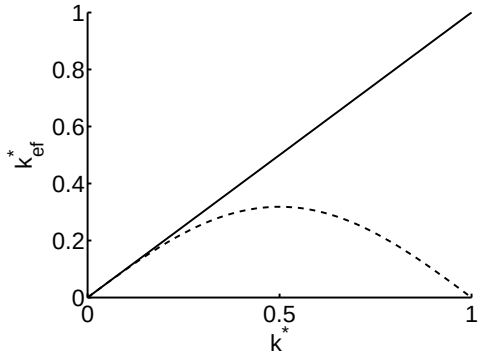


Figura 5.2: Número de onda modificada normalizada, k^*_{ef}, en función del número de onda normalizado, k^*. La línea continua es la solución exacta y la discontinua la correspondiente al esquema 5.29.

5.3 Estabilidad de la discretización espacio-temporal

El objetivo del análisis de estabilidad del esquema es determinar los valores que pueden tomar Δx y Δt en función de los parámetros del problema para que la solución numérica no sea divergente. Existen diferentes estrategias, aquí usaremos el *análisis de estabilidad de von Neumann*, que se aplica a problemas *lineales* en mallas *uniformes* en el espacio y con condiciones de contorno *periódicas*. El proceso es el siguiente:

1. Partimos de la ecuación lineal en derivadas parciales que queremos estudiar.

2. Realizamos la discretización temporal y espacial.

3. Consideramos una solución del tipo

$$u_i^n = r^n e^{Ikx_i}, \tag{5.32}$$

   donde I es la unidad imaginaria. Se trata de una onda periódica en el espacio con número de onda k y amplitud variable en el tiempo r^n (ver [1]), con r una constante.

4. Introducimos la solución anterior en la ecuación y usamos relaciones del tipo $x_{i+1} = x_i + \Delta x$ para eliminar el término e^{Ikx_i}.

[1] En este caso r^n significa r elevado a n, a diferencia de u_i^n que es la solución u en el instante t_n en el punto x_i

5. Para que la solución no sea divergente cuando $n \to \infty$ se debe cumplir $|r| < 1$ para todo k. Hay que tener en cuenta que r es un número complejo del tipo $a+Ib$, y que $|r| < 1$ significa $a^2+b^2 < 1$. Imponemos dicha condición y obtenemos la relación entre Δx y Δt para que la solución sea estable siempre (es decir, para el valor de k más crítico). A dicha relación se la suele denominar CFL (número de Courant-Friedrich-Levy).

Por ejemplo, consideremos la ecuación de difusión,

$$\frac{\partial u}{\partial t} = \nu \frac{\partial u^2}{\partial x^2}, \tag{5.33}$$

Su discretización espacial y temporal usando diferencias finitas centradas y Euler explícito es

$$u_i^{n+1} = u_i^n + \nu \Delta t/\Delta x^2 \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right). \tag{5.34}$$

Introducimos una solución del tipo $u_i^n = r^n e^{Ikx_i}$, y usando $x_{i+1} = x_i + \Delta x$ y $x_{i-1} = x_i - \Delta x$ eliminamos e^{Ikx_i}, lo que resulta

$$r = 1 + \nu \Delta t/\Delta x^2 \left(2\cos(k\Delta x) - 2\right). \tag{5.35}$$

Imponiendo $|r| < 1$ y para el caso más restrictivo de k (que es $\cos(k\Delta x) = -1$) obtenemos,

$$\Delta t < \Delta x^2/2\nu. \tag{5.36}$$

A la relación $2\Delta t\nu/\Delta x^2$ se la denomina CFL viscoso y se puede interpretar como el cociente entre el paso de tiempo numérico, Δt, y el tiempo característico de difusión viscosa, $\Delta x^2/\nu$. La condición (5.36) puede ser interpretada desde un punto de vista físico y nos indica que el paso de tiempo escogido debe ser menor que el tiempo característico de difusión para poder captar el fenómeno. Si consideramos la ecuación de convección,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \tag{5.37}$$

Que consiste en una onda que se desplaza con velocidad c a la derecha. Su discretización espacial y temporal usando diferencias finitas atrasadas y Euler explícito es

$$u_i^{n+1} = u_i^n - c\Delta t \left(u_i^n - u_{i-1}^n\right)/\Delta x. \tag{5.38}$$

Introducimos una solución del tipo $u_i^n = r^n e^{Ikx_i}$, y usando $x_{i-1} = x_i - \Delta x$, eliminamos e^{Ikx_i}, lo que resulta

$$r = 1 - c\Delta t/\Delta x \left(1 - cos(k\Delta x)\right) - Ic\Delta t/\Delta x \sin(k\Delta x) \tag{5.39}$$

Imponiendo $|r| < 1$ y para el caso más restrictivo de k obtenemos,

$$\Delta t < \Delta x/c. \tag{5.40}$$

A la relación $\Delta tc/\Delta x$ se la denomina CFL convectivo y se puede interpretar como el cociente entre el paso de tiempo numérico, Δt, y el tiempo característico de convección $\Delta x/c$. El esquema de diferencias finitas atrasadas produce resultados satisfactorios, pues capta bien el movimiento de la onda hacia la derecha. Usando un esquema de diferencias finitas adelantado o centrado se obtiene que nunca es estable. Cuando tratamos con ecuaciones no lineales, el método anterior no es válido. Lo que se suele hacer es aproximar el término $u\partial u/\partial x$ por $u_{max}\partial u/\partial x$, donde u_{max} es la máxima velocidad que existe en el flujo en un instante dado. Ésto linealiza el problema y permite *estimar* su estabilidad.
Según apliquemos el análisis de estabilidad a una ecuacion de convección o de difusión obtenemos CFLs que se denominan *CFL convectivo* y *CFL viscoso* respectivamente.

- CFL convectivo: impone una condición de la forma: $\Delta t < G(\Delta x, c)$.
- CFL viscoso: impone una condición de la forma: $\Delta t < G(\Delta x^2, \nu)$.

Donde G es una función que depende del esquema utilizado. Mirando las expresiones (5.36) y (5.40), podemos apreciar que cuando refinamos la malla (Δx más pequeño) la restricción viscosa es mucho más exigente puesto que depende de Δx^2. Estos esquemas se han obtenido usando discretización temporal explícita. Por esta razón, muchos esquemas numéricos discretizan los términos viscosos con esquemas implícitos que eliminan dicha inestabilidad. Cabe añadir que la relación CFL nos indica cuál debe ser Δt para que el esquema numérico no explote pero no indica cómo de buena es la solución, lo cual está relacionado con el error cometido.
El análisis de estabilidad se suele aplicar a los términos que tienen una discretización temporal explícita, ya que los que son discretizados implícitamente suelen ser condicionalmente estables para todo Δx y Δt tal y como se mencionó en el párrafo anterior. Cuando la malla no es uniforme, la experiencia nos dice que podemos hacer el análisis tomando el Δx_{max} y Δx_{min} del problema. También, en gran cantidad de problemas el carácter de estabilidad es independiente de las condiciones de contorno lo que justifica el uso de condiciones de contorno periódicas. Existe el llamado *análisis de estabilidad matricial*, que tiene en cuenta las condiciones de contorno y mallas no uniformes, sin embargo, no lo trataremos en estas notas.

5.4 Mallas de colocación y mallas staggered

Las diferencias finitas se suelen utilizar sobre todo con problemas discretizados espacialmente mediante mallas estructuradas (ver sección §2.2.2). Según cómo se organicen las variables (velocidades y presión) en la malla, se distingue entre *mallas de colocación* y *mallas staggered*.

- **Mallas de colocación**: todas las variables (u,v,w y p) se sitúan en los nodos de la malla y nunca se mueven ni a otro nodo ni a otro lugar de la malla. Por ello, las mallas de colocación son claras y sencillas de utilizar, además, permiten imponer fácilmente las condiciones de contorno dado que hay nodos situados directamente sobre el contorno. La figura 5.3(a) muestra un ejemplo de malla de colocación en 2D.

- **Mallas staggered**: cada variable (u,v,w y p) ocupa una posición diferente en la malla. Estas posiciones son las mismas al inicio y final del paso temporal, sin embargo, durante el proceso de cálculo de la solución en el instante t_{n+1} a partir de t_n, las variables así como sus sumas, productos y derivadas se mueven de un lugar a otro. Cada vez que sumamos o multiplicamos términos, tenemos que garantizar que están situados en los mismos puntos y no desplazados unos con respecto a las otros. Para ello, es necesario interpolar las variables según corresponda. La figura 5.3(b) muestra un ejemplo de malla staggered en 2D con cada variable situada en un lugar diferente de la malla. Las mallas staggered son ampliamente utilizadas con métodos de diferencias finitas y toda la complicación de mover variables es compensada por dos grandes **ventajas** (entre otras):

  1. *Permiten aumentar el orden de las diferencias finitas adelantadas y atrasadas.* Tal y como vimos en la sección §5.1, las diferencias finitas adelantadas y centradas son de la forma

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i \approx (u_{i+1} - u_i)/\Delta x + o(\Delta x), \tag{5.41}$$

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_i \approx (u_{i+1} - u_{i-1})/2\Delta x + o(\Delta x^2). \tag{5.42}$$

     Las diferencias finitas centradas son de orden dos, mientras que las adelantadas son de orden uno a pesar de utilizar la misma cantidad de puntos. El problema está en que estamos considerando que la derivada estimada con diferencias adelantadas está situada

en x_i cuando en realidad está en el punto medio entre x_i y x_{i+1} que denominaremos $x_{i+1/2}$. Es decir, podemos recuperar orden dos haciendo

$$\left(\widetilde{\frac{\partial u}{\partial x}}\right)_{i+1/2} \approx (u_{i+1} - u_i)/\Delta x + o(\Delta x^2). \tag{5.43}$$

$$\tag{5.44}$$

Como consecuencia, la variable se ha movido del punto x_i al punto $x_{i+1/2}$. En las mallas staggered, cada vez que derivamos una variable, ésta se mueve $\Delta x/2$ cuando derivamos en x y $\Delta y/2$ cuando lo hacemos en y.

2. *Acoplan las velocidades y presión garantizando que la solución sea única y evitando inestabilidades.* Para ilustrar esta característica pondremos como ejemplo la ecuación de ondas

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0. \tag{5.45}$$

Si consideramos el estado estacionario ($\partial u \partial t = 0$), la solución viene dada por $c\partial u/\partial x = 0$, que es $u = u_0$, con u_0 constante. Discreticemos la ecuación $c\partial u/\partial x = 0$ con diferencias finitas centradas,

$$c\frac{u_{i+1} - u_{i-1}}{2\Delta x} = 0, \tag{5.46}$$

cuya solución es $u_{i+1} = u_{i-1}$. Sin embargo, se puede apreciar rápidamente el siguiente inconveniente: los puntos x_i con i par son independientes de aquellos con i impar, por lo que podemos tener $u_{i,par} = u_1$ y $u_{i,impar} = u_2$ con u_1 y u_2 dos constantes diferentes. Ésto es fuente de inestabilidades y es común a todas las diferencias finitas centradas independientemente del orden o número de nodos que involucren. En 2D y 3D se vuelve aún peor pues aparecen más soluciones independientes. La solución se consigue usando diferencias finitas adelantadas o atrasadas en mallas staggered que acoplan todas las variables.

Por otro lado, como **inconvenientes** tenemos:

1. *Aumentan la complejidad del código.*
2. *Condiciones de contorno no se pueden imponer directamente.* En las mallas de colocación hay variables que están justo sobre el

(a) Malla de colocación (b) Malla staggered

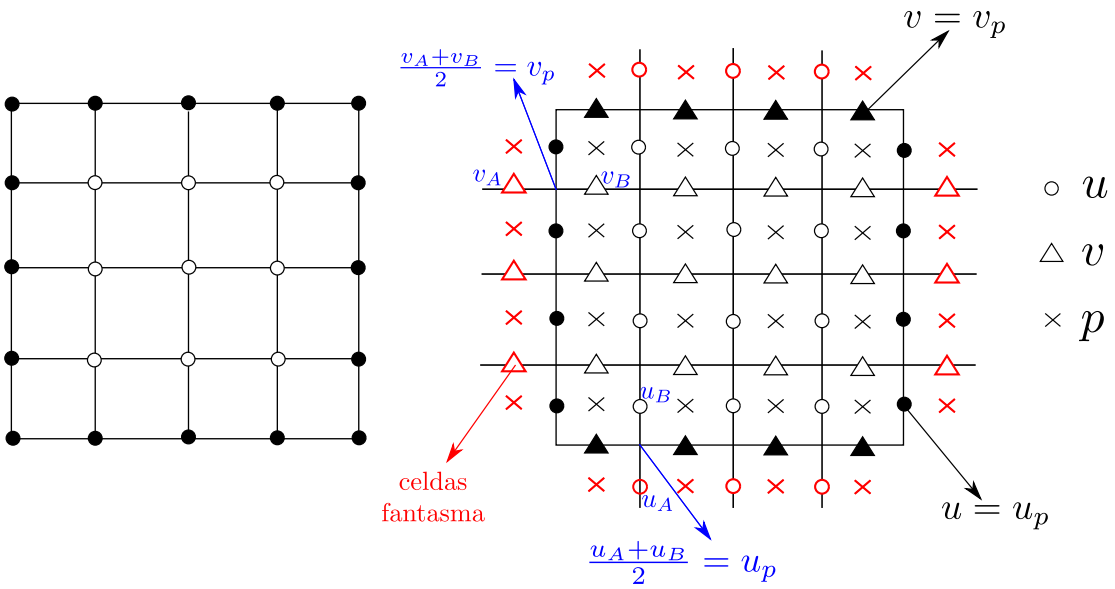


Figura 5.3: Ejemplo de mallas de colocación y staggered en 2D.

contorno (circulos en negrita en la figura 5.3(a)) y se usan directamente para imponer las condiciones de contorno. En las mallas staggered siempre falta alguna de variables sobre el contorno. Para solucionarlo se crean las llamadas *celdas fantasma* (símbolos en rojo en la figura 5.3(b)) que pasan a ser nuevas incógnitas. A cambio, se añade la condición de que el valor interpolado en la frontera sea la condición de contorno correspondiente.

5.5 Aplicación a problemas 1D

En esta sección utilizaremos los conceptos de discretización espacial y temporal estudiados hasta ahora para resolver los problemas tipo unidimensionales: la ecuación del calor, la de onda y la de Burgers viscosa. Solamente haremos uso de algunos esquemas y se deja como ejercicio al lector que resuelva los problemas aplicando otros. También aprovecharemos estos ejemplos para explicar cómo se tratan las condiciones de contorno y la organización matricial de las ecuaciones discretizadas.
Denotaremos la solución exacta como $u(x,t)$, la discretización temporal como $u^n(x)$ en $t = t_n$, la espacial como $u_i(t)$ en $x = x_i$ y ambas a la vez u_i^n.

Consideraremos $0 \leq x \leq 1$ y una malla de colocación uniforme con $N+1$ puntos x_i, $i = 0, .., N$ y $x_{i+1} - x_i = \Delta x$.

5.5.1 Ecuación de onda

Queremos resolver la ecuación hiperbólica

$$\frac{\partial u(x,t)}{\partial t} + c\frac{\partial u(x,t)}{\partial x} = 0, \tag{5.47}$$

con $c > 0$, para $0 \leq x \leq 1$ con la condición de contorno

$$u(0,t) = u_0 \tag{5.48}$$

y condición inicial $u(x,0) = u_I(x)$, compatible con la condición de contorno anterior. Para la discretización temporal utilizaremos el esquema *Euler explícito*, lo que resulta

$$u^{n+1} = u^n - \Delta t c\frac{\partial u^n}{\partial x}. \tag{5.49}$$

Dado que para $c > 0$, la ecuación (5.48) representa la convección de la condición inicial $u_I(x)$ hacia la derecha, para la derivada espacial utilizaremos *diferencias finitas atrasadas primer orden*:

$$\frac{\partial u_i}{\partial x} \approx \frac{u_i - u_{i-1}}{\Delta x}. \tag{5.50}$$

Se deja como ejercicio probar qué ocurre si utilizamos diferencias finitas adelantadas o centradas cuando $c > 0$. Combinando los resultados anteriores obtenemos el sistema de ecuaciones que hay que resolver:

$$\begin{aligned} u_i^{n+1} &= u_i^n - \Delta t c\frac{u_i^n - u_{i-1}^n}{\Delta x}, \quad i = 1, .., N, \\ u_0^n &= u_0. \end{aligned} \tag{5.51}$$

Las ecuaciones (5.51) tienen como incógnitas $u_1^{n+1}, .., u_N^{n+1}$ y pueden ser organizadas de forma matricial

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^{n+1} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^{n} - \frac{\Delta t c}{\Delta x} \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ \vdots & & & & & \\ 0 & \dots & 0 & -1 & 1 & 0 \\ 0 & \dots & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^{n} - \frac{\Delta t c}{\Delta x} \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ \\ 0 \end{pmatrix}^{n} \tag{5.52}$$

donde el último vector proviene de las condiciones de contorno y aparece de forma natural al reorganizar (5.51) . El sistema de ecuaciones se puede escribir de forma más compacta

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \frac{\Delta t c}{\Delta x} D_x \boldsymbol{u}^n - \frac{\Delta t c}{\Delta x} \boldsymbol{u}_{cc}, \tag{5.53}$$

donde

$$\boldsymbol{u}^n = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^n , \tag{5.54}$$

D_x es la matriz de diferenciación para la derivada primera y $\boldsymbol{u}_{cc}$ el vector que proviene de las condiciones de contorno.
A continuación se muestra el código en *Matlab*© para resolver el sistema (5.53):
(Archivo: ec_onda.m)

```
%---------------------------------------%
% Ecuacion de onda con C.C. Dirichlet   %
%                                       %
% ut + u ux = 0                         %
% u(0,t) = u0                           %
%                                       %
% Discretizacion temporal:              %
% Euler explicito                       %
%                                       %
% Discretizacion espacial:              %
% diferencias finitas atrasadas         %
%                                       %
% A. Lozano Duran                       %
% 2014                                  %
%---------------------------------------%

disp('Ecuacion de onda')

clear all

% Parametros:
c   = 1;   % velocidad de conveccion
CFL = 0.5; % CFL
Lx  = 1;   % longitud del dominio
nx  = 100; % numero de puntos interiores
Nt  = 300; % numero de pasos temporales

dx  = Lx/nx;
x   = dx*(0:nx);
```

```
disp(['c:   ',num2str(Lx)])
disp(['Lx:  ',num2str(Lx)])
disp(['nx:  ',num2str(nx)])
disp(['CFL: ',num2str(CFL)])

% Condicion inicial:
ui = sin(4*pi*x)';
u  = ui(2:nx+1);

% Condicion de contorno en x=0
u0     = ui(1);

uc     = zeros(nx,1);
uc(1)  = u0;

% Matrices de diferenciacion
v  = ones(nx+2,1);
% Derivada primera: upwind
Bx  = sparse(nx+2,nx+2);
Bx  = 1/dx*(spdiags(-1*v,-1,Bx) + spdiags(1*v,0,Bx));
Dx  = Bx(2:end-1,2:end-1);
Dx0 = Bx(2:end-1,1);

% Paso de tiempo convectivo
dt = CFL*dx/c;

% Inicio
t = 0;
for i=1:Nt

    % Avanzar en el tiempo: Euler explicito
    u = u  - dt*c*Dx*u - dt*c*uc;
    t = t + dt;

    % Pintar solucion
    u_numerica = [u0 u'];
    u_exacta   = sin(4*pi*(x-c*t))'; u_exacta(1:ceil(c*t*nx)) = 0;
    figure(1),set(gca,'LineWidth',2,'fontsize',22,'fontname','times')
    plot(x,u_numerica,'b',x,u_exacta,'r--','LineWidth',2)
    axis([0 1 -1 1])
```

```
    xlabel('x'),ylabel('u')
    legend('Numerica','Exacta')
    drawnow

end

disp('Fin')
%--------------------------------------%
```

5.5.2 Ecuación del calor

Queremos resolver la ecuación parabólica

$$\frac{\partial u(x,t)}{\partial t} = \nu \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{5.55}$$

con $\nu > 0$, para $0 \leq x \leq 1$ con condiciones de contorno

$$u(0,t) = u_0, \quad u(1,t) = u_N, \tag{5.56}$$

y condición inicial $u(x,0) = u_I(x)$, compatible con las condiciones de contorno anteriores. Para la discretización temporal utilizaremos el esquema *Euler implícito*, lo que resulta

$$u^{n+1} = u^n + \Delta t \nu \frac{\partial^2 u^{n+1}}{\partial x^2}. \tag{5.57}$$

Para la derivada espacial utilizaremos *diferencias finitas centradas de segundo orden*:

$$\frac{\partial^2 u_i}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}. \tag{5.58}$$

Combinando los resultados anteriores obtenemos el sistema de ecuaciones que hay que resolver:

$$\begin{aligned} u_i^{n+1} &= u_i^n + \Delta t \nu \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}, \quad i = 1, .., N-1, \\ u_0^{n+1} &= u_0, \quad u_N^{n+1} = u_N. \end{aligned} \tag{5.59}$$

Las ecuaciones (5.59) tienen como incógnitas $u_1^{n+1}, .., u_{N-1}^{n+1}$ y pueden ser organizadas de forma matricial

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}^{n+1} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}^{n} + \frac{\Delta t \nu}{\Delta x^2} \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & & & & & \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & \dots & 0 & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}^{n+1} + \frac{\Delta t \nu}{\Delta x^2} \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{pmatrix} \tag{5.60}$$

donde el último vector proviene de las condiciones de contorno. El sistema de ecuaciones se puede escribir de forma más compacta

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t\nu}{\Delta x^2} D_{xx}\boldsymbol{u}^{n+1} + \frac{\Delta t\nu}{\Delta x^2}\boldsymbol{u}_{cc}, \tag{5.61}$$

con D_{xx} la matriz de diferenciación para la derivada segunda y $\boldsymbol{u}_{cc}$ el vector que proviene de las condiciones de contorno. Despejando $\boldsymbol{u}^{n+1}$:

$$\left(\mathbb{I} - \frac{\Delta t\nu}{\Delta x^2} D_{xx}\right)\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t\nu}{\Delta x^2}\boldsymbol{u}_{cc}, \tag{5.62}$$

con $\mathbb{I}$ la matrix identidad. Para resolver el sistema (5.62) es necesario invertir la matriz $(\mathbb{I} - \Delta t\nu/\Delta x^2 D_{xx})$.
El problema (5.55) tiene solución analítica cuando las condiciones de contorno son homogéneas ($u_0 = u_N = 0$):

$$u(x,t) = \sum_{n=1}^{\infty}\left(2\int_0^1 u_I(y)\sin(\pi n y)\mathrm{d}y\right)\sin(\pi n x)\exp(-\nu\pi^2 n^2 t) \tag{5.63}$$

que puede ser utilizada para validar la solución numérica.
A continuación se muestra el código en *Matlab*© para resolver el sistema (5.62):
(Archivo: ec_calor.m)

```
%---------------------------------------%
% Ecuacion del calor con C.C. Dirichlet %
%                                       %
% ut = nu uxx                           %
% u(0,t) = u0                           %
% u(1,t) = uN                           %
%                                       %
% Discretizacion temporal:              %
% Euler implicito                       %
%                                       %
% Discretizacion espacial:              %
% diferencias finitas centradas         %
%                                       %
% A. Lozano Duran                       %
% 2014                                  %
%---------------------------------------%

disp('Ecuacion del calor')
```

```
clear all

% Parametros:
nu =  0.1;  % viscosidad
CFL = 2;    % CFL
Lx  = 1;    % longitud del dominio
nx  = 100;  % numero de puntos interiores
Nt  = 1200; % numero de pasos temporales

dx  = Lx/(nx+1);
x   = dx*(0:nx+1);

disp(['nu:  ',num2str(nu)])
disp(['Lx:  ',num2str(Lx)])
disp(['nx:  ',num2str(nx)])
disp(['CFL: ',num2str(CFL)])

% Condicion inicial:
ui = sin(4*pi*x)';
u  = ui(2:nx+1);

% Condiciones de contorno:
u0     = ui(1);     % en x=0
uN     = ui(nx+2);  % en x=1

uc     = zeros(nx,1);
uc(1)  = u0;
uc(nx) = uN;

% Matrices de diferenciacion
v  = ones(nx+2,1);
% Derivada segunda: centrada
Bxx  = sparse(nx+2,nx+2);
Bxx  = 1/dx^2*(spdiags(1*v,1,Bxx) + spdiags(-2*v,0,Bxx) ...
           + spdiags(1*v,-1,Bxx));
Dxx  = Bxx(2:end-1,2:end-1);
Dxx0 = Bxx(2:end-1,1);
DxxN = Bxx(2:end-1,nx+2);
% Matriz identidad
I    = eye(nx,nx);
```

```
% Paso de tiempo viscoso
dt = CFL*dx^2/nu;

% Inicio
t = 0;
for i=1:Nt

    % Avanzar en el tiempo: Euler implicito A*u=b
    A = (I-dt*nu*Dxx);
    b = u + dt*nu*uc;
    u = A\b;
    t = t + dt;

    % Pintar solucion
    u_numerica = [u0 u' uN];
    solucion_calor;
    figure(1),set(gca,'LineWidth',2,'fontsize',22','fontname','times')
    plot(x,u_numerica,'b',x,u_exacta,'r--','LineWidth',2)
    axis([0 1 -1 1])
    xlabel('x'),ylabel('u')
    legend('Numerica','Exacta')
    drawnow

end

disp('Fin')
%---------------------------------------%
```

(Archivo: solucion_calor.m)

```
%---------------------------------------%
% solucion analitica ecuacion del calor  %
%---------------------------------------%

if i==1,
    nm = 20;
    for j=1:nm
        a(j) = 2*trapz(x,ui.*sin(pi*j*x'));
    end
end
```

```
u_exacta = zeros(nx+2,1);
for j=1:nm
   u_exacta = u_exacta + a(j)*sin(pi*j*x')*exp(-nu*pi^2*j^2*t);
end
```

5.5.3 Ecuación de Burgers viscosa

Planteamos ahora la ecuación de Burgers viscosa

$$\frac{\partial u(x,t)}{\partial t} + u\frac{\partial u(x,t)}{\partial x} = \nu\frac{\partial^2 u(x,t)}{\partial x^2}, \tag{5.64}$$

con $c > 0$, $\nu > 0$, para $0 \leq x \leq 1$ con condiciones de contorno

$$u(0,t) = u_0, \quad u(1,t) = u_N, \tag{5.65}$$

y condición inicial $u(x,0) = u_I(x)$, compatible con las condiciones de contorno anteriores. Utilizaremos la ecuación (5.64) de la forma

$$\frac{\partial u(x,t)}{\partial t} + 1/2\frac{\partial u^2(x,t)}{\partial x} = \nu\frac{\partial^2 u(x,t)}{\partial x^2}, \tag{5.66}$$

Para la discretización temporal usaremos el esquema *Runge-Kutta de 4 pasos*, lo que resulta

$$\begin{aligned}
u^{n+1} &= u^n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), && (5.67)\\
k_1 &= F(u^n), && (5.68)\\
k_2 &= F(u^n + k_1\Delta t/2), && (5.69)\\
k_3 &= F(u^n + k_2\Delta t/2), && (5.70)\\
k_4 &= F(u^n + k_3\Delta t), && (5.71)\\
F(u) &= -\Delta t\frac{\partial u^{2n}/2}{\partial x} + \Delta t\nu\frac{\partial^2 u^n}{\partial x^2}. && (5.72)
\end{aligned}$$

Para la derivada espacial primera utilizaremos *diferencias finitas atrasadas de primer orden*:

$$\frac{\partial u_i^2}{\partial x} \approx \frac{u_i^2 - u_{i-1}^2}{\Delta x}, \tag{5.73}$$

y para la derivada segunda *diferencias finitas centradas de segundo orden*:

$$\frac{\partial^2 u_i}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}, \tag{5.74}$$

Combinando los resultados anteriores obtenemos el sistema de ecuaciones que hay que resolver:

$$
\begin{aligned}
u_i^{n+1} &= u_i^n + \frac{\Delta t}{6}\left(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}\right), \qquad (5.75)\\
k_{1i} &= F(u_i^n),\\
k_{2i} &= F(u_i^n + k_{1i}\Delta t/2),\\
k_{3i} &= F(u_i^n + k_{2i}\Delta t/2),\\
k_{4i} &= F(u_i^n + k_{3i}\Delta t),\\
F(u_i^n) &= -\Delta t/2\frac{u_i^{2n} - u_{i-1}^{2n}}{\Delta x} + \Delta t\nu\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \quad i = 1, .., N-1,\\
u_0^{n+1} &= u_0, \quad u_N^{n+1} = u_N. \qquad (5.76)
\end{aligned}
$$

Las ecuaciones (5.75) tienen como incógnitas $u_1^{n+1}, .., u_{N-1}^{n+1}$ y pueden ser organizadas de forma matricial

$$
\begin{aligned}
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^n + \frac{\Delta t}{6}\left(\boldsymbol{k_1} + 2\boldsymbol{k_2} + \boldsymbol{2k_3} + \boldsymbol{k_4}\right), \qquad (5.77)\\
\boldsymbol{k_1} &= \boldsymbol{F}(\boldsymbol{u}^n),\\
\boldsymbol{k_2} &= \boldsymbol{F}(\boldsymbol{u}^n + \boldsymbol{k_1}\Delta t/2),\\
\boldsymbol{k_3} &= \boldsymbol{F}(\boldsymbol{u}^n + \boldsymbol{k_2}\Delta t/2),\\
\boldsymbol{k_4} &= \boldsymbol{F}(\boldsymbol{u}^n + \boldsymbol{k_3}\Delta t),\\
\boldsymbol{F}(\boldsymbol{u}^n) &= -\frac{\Delta t}{2\Delta x}D_x\boldsymbol{u}^{2n} + \frac{\Delta t\nu}{\Delta x^2}D_{xx}\boldsymbol{u}^n - \frac{\Delta t}{2\Delta x}\boldsymbol{u^2}_{cc1} + \frac{\Delta t\nu}{\Delta x^2}\boldsymbol{u}_{cc2},
\end{aligned}
$$

con D_x la matriz de diferenciación para la derivada primera, D_{xx} la matriz de diferenciación para la derivada segunda, $\boldsymbol{u^2}_{cc1}$ y $\boldsymbol{u}_{cc2}$ los vectores que procedentes de las condiciones de contorno y

$$
\boldsymbol{u}^n = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}^n, \quad \boldsymbol{k_j} = \begin{pmatrix} k_{j1} \\ k_{j2} \\ \vdots \\ k_{jN-2} \\ k_{jN-1} \end{pmatrix} \qquad (5.78)
$$

El término $\boldsymbol{F}(\boldsymbol{u}^n)$ desarrollado es

$$\begin{aligned}\boldsymbol{F}(\boldsymbol{u}^n) &= -\frac{\Delta t}{2\Delta x}\begin{pmatrix} 1 & 0 & 0 & 0 & ... & 0 \\ -1 & 1 & 0 & 0 & ... & 0 \\ 0 & -1 & 1 & 0 & ... & 0 \\ \vdots & ... & 0 & -1 & 1 & 0 \\ 0 & ... & 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} u_1^2 \\ u_2^2 \\ \vdots \\ u_{N-2}^2 \\ u_{N-1}^2 \end{pmatrix}^n \\ &+ \frac{\Delta t\nu}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & 0 & ... & 0 \\ 1 & -2 & 1 & 0 & ... & 0 \\ 0 & 1 & -2 & 1 & ... & 0 \\ \vdots & ... & 0 & 1 & -2 & 1 \\ 0 & ... & 0 & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}^n - \frac{\Delta t}{2\Delta x}\begin{pmatrix} u_0^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \frac{\Delta t\nu}{\Delta x^2}\begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{pmatrix}.\end{aligned} \tag{5.79}$$

El problema (5.64) tiene solución analítica cuando las condiciones de contorno son homogéneas ($u_0 = u_N = 0$):

$$u(x,t) = 2\pi\nu\frac{\sum_{n=1}^{\infty} a_n \exp(-n^2\pi^2\nu t) n \sin(n\pi x)}{a_0 + \sum_{n=1}^{\infty} a_n \exp(-n^2\pi^2\nu t)\cos(n\pi x)}, \tag{5.80}$$

$$a_0 = \int_0^1 \exp(-x^2/(3\nu)(3-2x))\mathrm{d}x, \tag{5.81}$$

$$a_n = \int_0^1 \exp(-x^2/(3\nu)(3-2x))\cos(n\pi x)\mathrm{d}x, \quad n = 1, 2... \tag{5.82}$$

$$\tag{5.83}$$

con la condición inicial $u_I(x) = 4x(1-x)$, que puede ser utilizada para validar la solución numérica.
A continuación se muestra el código en *Matlab*© para resolver el sistema (5.77):
(Archivo: ec_burgers.m)

```
%-----------------------------------------%
% Ecuacion de burgers con C.C. Dirichlet %
%                                         %
% ut + 1/2(u^2)x = nu uxx                 %
% u(0,t) = u0                             %
% u(1,t) = uN                             %
%                                         %
% Discretizacion temporal:                %
% Runge-Kutta 4 pasos                     %
%                                         %
% Discretizacion espacial:                %
```

```
% diferencias finitas atrasadas para dx  %
% diferencias finitas centradas para dxx %
%                                        %
% A. Lozano Duran                        %
% 2014                                   %
%----------------------------------------%

disp('Ecuacion de Burgers viscosa')

clear all

global nu Dx Dx0 DxN Dxx Dxx0 DxxN u0 uN

% Parametros:
nu =  0.02; % viscosidad
CFL = 0.5;  % CFL
Lx  = 1;    % longitud del dominio
nx  = 100;  % numero de puntos interiores
Nt  = 200;  % numero de pasos temporales

dx  = Lx/(nx+1);
x   = dx*(0:nx+1);

disp(['nu:  ',num2str(nu)])
disp(['Lx:  ',num2str(Lx)])
disp(['nx:  ',num2str(nx)])
disp(['CFL: ',num2str(CFL)])

% Condicion inicial:
ui = 4*x.*(1-x); ui=ui';
u  = ui(2:nx+1);

% Condiciones de contorno:
u0     = ui(1);     % en x=0
uN     = ui(nx+2);  % en x=1

uc     = zeros(nx,1);
uc(1)  = u0;
uc(nx) = uN;

% Matrices de diferenciacion
```

```
v    = ones(nx+2,1);
% Derivada primera: upwind
Bx   = sparse(nx+2,nx+2);
Bx   = 1/dx*(spdiags(-1*v,-1,Bx) + spdiags(1*v,0,Bx));
Dx   = Bx(2:end-1,2:end-1);
Dx0  = Bx(2:end-1,1);
DxN  = Bx(2:end-1,nx+2);
% Derivada segunda: centrada
Bxx  = sparse(nx+2,nx+2);
Bxx  = 1/dx^2*(spdiags(1*v,1,Bxx) + spdiags(-2*v,0,Bxx) ...
       + spdiags(1*v,-1,Bxx));
Dxx  = Bxx(2:end-1,2:end-1);
Dxx0 = Bxx(2:end-1,1);
DxxN = Bxx(2:end-1,nx+2);

% Inicio
t = 0;
for i=1:Nt

    % Paso de tiempo convectivo
    dtc = CFL*dx/max(u);

    % Paso de tiempo viscoso
    dtv = CFL*dx^2/nu;

    % Paso de tiempo:
    dt = min(dtc,dtv);

    % Avanzar en el tiempo: Runge-kutta 4
    k1 = rhs(u);
    k2 = rhs(u+1/2*k1*dt);
    k3 = rhs(u+1/2*k2*dt);
    k4 = rhs(u+k3*dt);
    u  = u + 1/6*dt*( k1 + 2*k2 + 2*k3 + k4 );
    t = t + dt;

    % Pintar solucion
    u_numerica = [u0 u' uN];
    solucion_burgers;
    figure(1),set(gca,'LineWidth',2,'fontsize',22','fontname','times')
    plot(x,u_numerica,'b',x,u_exacta,'r--','LineWidth',2)
```

```
    axis([0 1 0 1])
    xlabel('x'),ylabel('u')
    legend('Numerica','Exacta')
    drawnow

end

disp('Fin')
%---------------------------------------%
```

(Archivo: rhs.m)

```
%---------------------------------------%
% funcion F = - u^2xx + nu uxx          %
%---------------------------------------%
function v = rhs(u)

global nu Dx Dx0 DxN Dxx Dxx0 DxxN u0 uN

v =  -Dx*(u.^2/2) + nu*Dxx*u    ...
     -Dx0*u0.^2/2 + nu*Dxx0*u0  ... % x=0
     -DxN*uN.^2/2 + nu*DxxN*uN;     % x=L

end
```

(Archivo: solucion_burgers.m)

```
%---------------------------------------%
% calcula la solucion 'exacta' ec. burgers %
%---------------------------------------%

if i==1,
    nm = 100;
    an(1) = trapz(x, exp(-x.^2/(3*nu).*(3-2*x)) );
    for jj=2:nm
        an(jj) = 2*trapz(x, exp(-x.^2/(3*nu).*(3-2*x)).*cos((jj-1)*pi*x) );
    end
end

I1 = zeros(1,length(x));
```

```
I2 = zeros(1,length(x));
for jj=1:nm-1
    I1 = I1 + an(jj+1)*exp(-jj^2*pi^2*nu*t)*jj*sin(jj*pi*x);
    I2 = I2 + an(jj+1)*exp(-jj^2*pi^2*nu*t)*cos(jj*pi*x);
end
I2 = I2 + an(1);
u_exacta = 2*pi*nu*I1./I2;
```

5.6 Aplicación a problemas 2D: la cavidad

En este apartado resolveremos el problema de la cavidad en dos dimensiones. Se trata de un problema muy completo que contiene elementos fundamentales como el uso de mallas staggered, el paso temporal implícito para el término viscoso y el paso fraccionado para conseguir la incompresibilidad del fluido. Es importante entender los casos unidimensionales del apartado anterior para poder seguir el código bidimensional. Aunque existen gran cantidad de detalles que por brevedad no se comentan, a continuación explicamos los aspectos clave del proceso.

5.6.1 Planteamiento del problema

Consideraremos un fluido bidimensional confinado dentro de una cavidad cuadrada cuya pared superior se mueva con velocidad $u = 1$ y $v = 0$, donde u y v son las velocidades en x e y respectivamente. La figura 5.4 muestra un esquema del problema. Resolveremos las ecuaciones de Navier-Stokes en 2D

$$\frac{\partial u}{\partial t} = -\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{5.84}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial v^2}{\partial y} - \frac{\partial uv}{\partial x} - \frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{5.85}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{5.86}$$

para $0 < x < 1$ y $0 < y < 1$, junto con las condiciones de contorno

$$u(x,1) = 1, \quad v(x,1) = 0, u(x,0) = 0, \quad v(x,0) = 0, \tag{5.87}$$

$$u(1,y) = 0, \quad v(1,y) = 0, u(0,y) = 0, \quad v(0,y) = 0. \tag{5.88}$$

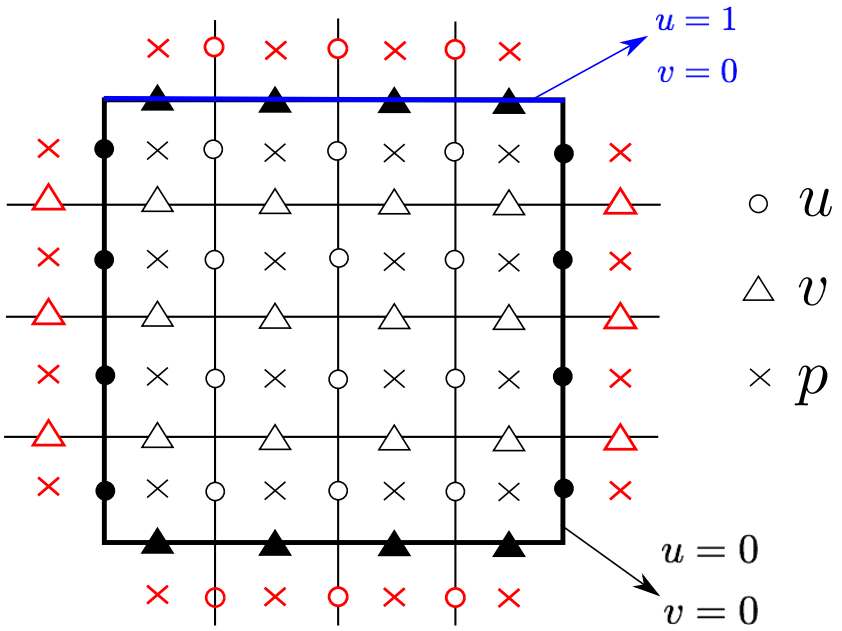


Figura 5.4: Geometría y condiciones de contorno de la cavidad y organización de las variables en la malla stagggered.

5.6.2 Discretización temporal

Utilizaremos un esquema *Euler explícito* para discretizar los términos no lineales, *Euler implícito* para los viscosos y *paso fraccionado* para forzar la incompresibilidad. Las ecuaciones que resultan son

$$\begin{aligned}
u^* &= u^n + \Delta t \left(-\frac{\partial u^{2n}}{\partial x} - \frac{\partial u^n v^n}{\partial y} + \nu \left(\frac{\partial^2 u^*}{\partial x^2} + \frac{\partial^2 u^*}{\partial y^2} \right) \right), \\
v^* &= v^n + \Delta t \left(-\frac{\partial v^{2n}}{\partial y} - \frac{\partial u^n v^n}{\partial x} + \nu \left(\frac{\partial^2 v^*}{\partial x^2} + \frac{\partial^2 v^*}{\partial y^2} \right) \right), \\
\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} &= \frac{1}{\Delta t} \left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y} \right), \\
u^{n+1} &= u^* - \Delta t \frac{\partial p}{\partial x}, \\
v^{n+1} &= v^* - \Delta t \frac{\partial p}{\partial y}.
\end{aligned} \tag{5.89}$$

5.6.3 Discretización espacial

Utilizaremos una *malla staggered* como la que se muestra en la figura 5.4 con nodos x_i, $i = 1, .., n_x$, y_j, $j = 1, .., n_y$ y uniformemente espaciada con $x_{i+1} -$

$x_i = \Delta x$ y $y_{j+1} - y_j = \Delta y$. El número de puntos n_x y n_y se corresponde con la cantidad de puntos interiores al dominio, y se utilizarán celdas fantasma para imponer las condiciones de contorno. Para la derivada primera y segunda usaremos *diferencias finitas centradas*

$$\begin{aligned}
\frac{\partial u_{i+1/2,j}}{\partial x} &\approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \\
\frac{\partial u_{i,j+1/2}}{\partial y} &\approx \frac{u_{i,j+1} - u_{i,j}}{\Delta y}, \\
\frac{\partial^2 u_{i,j}}{\partial x^2} &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \\
\frac{\partial^2 u_{i,j}}{\partial y^2} &\approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta x^2},
\end{aligned}$$

y lo mismo para las derivadas de v. Hay que tener en cuenta que cada vez que calculemos la derivada primera de una variable, ésta cambiará su posición en la malla. Sin embargo con la derivada segunda mantendrá su posición. Por ello es importante que cada vez que sumemos dos términos éstos, tengan la misma posición en la malla. En caso contrario, será necesario interpolar uno de ellos para colocarlo en la posición adecuada. Utilizaremos interpolación lineal, por ejemplo, para mover $u_{i,j}$ a $u_{i,j+1/2}$ haremos

$$u_{i,j+1/2} = \frac{u_{i,j+1} + u_{i,j}}{2}. \tag{5.90}$$

Tanto al principio como al final de cada paso temporal, las variables u, v y p ocuparán las posiciones que se muestran en la figura 5.4. Ésto hace que las variables u, v y p tengan dimensiones distintas. La siguiente tabla muestra las dimensiones de cada variable según consideremos sus puntos interiores o interiores+exteriores:

	puntos interiores	puntos interiores+exteriores
u	$(n_x - 1) \times n_y$	$(n_x + 1) \times (n_y + 2)$
v	$n_x \times (n_y - 1)$	$(n_x + 2) \times (n_y + 1)$
p	$n_x \times n_y$	$(n_x + 2) \times (n_y + 2)$

La ecuaciones discretizadas que resultan para u y p son

$$u^*_{i+1/2,j} = u^n_{i+1/2,j} + \Delta t \left(-\frac{u^{2n}_{i+1,j} - u^{2n}_{i,j}}{\Delta x} - \frac{(u^n v^n)_{i+1/2,j+1} - (u^n v^n)_{i+1/2,j}}{\Delta y} \right) \quad (5.91)$$
$$+\Delta t \nu \left(\frac{u^*_{i+1/2+1,j} - 2u^*_{i+1/2,j} + u^*_{i+1/2-1,j}}{\Delta x^2} + \frac{u^*_{i+1/2,j+1} - 2u^*_{i+1/2,j} + u^*_{i+1/2,j-1}}{\Delta y^2} \right),$$

$$\frac{p^{n+1}_{i+1,j} - 2p^{n+1}_{i,j} + p^{n+1}_{i-1,j}}{\Delta x^2} + \frac{p^{n+1}_{i,j+1} - 2p^{n+1}_{i,j} + p^{n+1}_{i,j-1}}{\Delta y^2} = \quad (5.92)$$
$$= \frac{1}{\Delta t} \left(\frac{u^*_{i+1/2,j} - u^*_{i-1/2,j}}{\Delta x} + \frac{v^*_{i,j+1/2} - v^*_{i,j-1/2}}{\Delta y} \right),$$

$$u^{n+1}_{i+1/2,j} = u^*_{i+1/2,j} - \Delta t \frac{p^{n+1}_{i+1,j} - p^{n+1}_{i,j}}{\Delta x}. \quad (5.93)$$

Se deja como ejercicio al lector obtener las ecuaciones para v.

5.6.4 Organización matricial de las ecuaciones

Al igual que se hizo en la sección §5.5 con los ejemplos unidimensionales, las ecuaciones (5.91) se pueden organizar de forma matricial. Para ello, tenemos que decidir cómo organizar las variables $u^n_{i+1/2,j}$, $v^n_{i,j+1/2}$ y $p^{n+1}_{i,j}$ en vectores columna $\boldsymbol{u}^n$, $\boldsymbol{v}^n$ y $\boldsymbol{p}^{n+1}$. En nuestro caso lo haremos colocando en un vector las columnas de $u_{i,j}$ y lo mismo para el resto de variables, aunque otras formas son igualmente válidas. Como resultado tenemos

$$\boldsymbol{u}^n = \begin{pmatrix} u_{1+1/2,1} \\ u_{1+1/2,2} \\ \vdots \\ u_{1+1/2,ny} \\ u_{2+1/2,1} \\ u_{2+1/2,2} \\ \vdots \\ u_{n_x-1+1/2,n_y} \end{pmatrix}^n, \boldsymbol{v}^n = \begin{pmatrix} v_{1,1+1/2} \\ v_{1,2+1/2} \\ \vdots \\ v_{1,n_y-1+1/2} \\ v_{2,1+1/2} \\ v_{2,2+1/2} \\ \vdots \\ v_{n_x,n_y-1+1/2} \end{pmatrix}^n, \boldsymbol{p}^{n+1} = \begin{pmatrix} p_{1,1} \\ p_{1,2} \\ \vdots \\ p_{1,n_y} \\ p_{2,1} \\ p_{2,2} \\ \vdots \\ p_{n_x,n_y} \end{pmatrix}^{n+1} \quad (5.94)$$

De esta forma, las derivadas primeras y segundas se pueden expresar como el producto de las matrices D_x (derivada primera) y D_{xx} (derivada segunda) por los vectores $\boldsymbol{u}^n$ y $\boldsymbol{v}^n$ o $\boldsymbol{u}^{n+1}$ y $\boldsymbol{v}^{n+1}$ según corresponda. En este caso nos interesan las derivadas segundas, pues aparecen en el término implícito y necesitamos su forma matricial para poder resolver el sistema. Las derivadas primeras las calcularemos directamente sin usar matrices. La forma de las matrices D_{xx} y D_{yy} se puede deducir a partir de las ecuaciones (5.91), sin

embargo, existe un procedimiento más automático utilizando el *producto de Kronecker*, $\otimes$, el cual transforma las matrices de diferenciación 1D utilizadas en el apartado §5.5 en matrices de diferenciación 2D. De esta forma, si D_{xx}^{1D} es la matriz de diferenciación obtenida para los casos 1D, tenemos que

$$\begin{aligned} D_{xx} &= \mathbb{I} \otimes D_{xx}^{1D}, \\ D_{yy} &= D_{xx}^{1D} \otimes \mathbb{I}, \end{aligned} \tag{5.95}$$

donde $\mathbb{I}$ es la matriz identidad con las dimensiones adecuadas según la variables u, v o p. que queremos derivar. El orden en el cual se realiza el producto está relacionado con la forma de organizar las variables $u_{i+1/2,j}$ por filas o por columnas. Dado que u, v y p tiene dimensiones diferentes, tendremos varias matrices D_{xx} y D_{yy} asociadas a cada una de ellas y que denotaremos como D_{xx}^{u}, D_{xx}^{v} y D_{xx}^{p}. Lo mismo se aplica para D_{yy}. Las ecuaciones en notación matricial son

$$\left(\mathbb{I} - \frac{\nu\Delta t}{\Delta x^2} D_{xx}^{u} - \frac{\nu\Delta t}{\Delta y^2} D_{yy}^{u}\right) \boldsymbol{u}^{*} = \boldsymbol{u}^{n} + \Delta t \boldsymbol{NL}_{u}^{n}, \tag{5.96}$$

$$\left(\mathbb{I} - \frac{\nu\Delta t}{\Delta x^2} D_{xx}^{v} - \frac{\nu\Delta t}{\Delta y^2} D_{yy}^{v}\right) \boldsymbol{v}^{*} = \boldsymbol{v}^{n} + \Delta t \boldsymbol{NL}_{v}^{n}, \tag{5.97}$$

$$\left(\frac{1}{\Delta x^2} D_{xx}^{p} + \frac{1}{\Delta y^2} D_{yy}^{p}\right) \boldsymbol{p}^{n+1} = \frac{1}{\Delta t}\left(\delta_x \boldsymbol{u}^{*} + \delta_y \boldsymbol{v}^{*}\right), \tag{5.98}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{*} - \Delta t \delta_x \boldsymbol{p}^{n+1}, \tag{5.99}$$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^{*} - \Delta t \delta_y \boldsymbol{p}^{n+1}, \tag{5.100}$$

donde $\boldsymbol{NL}_{u}^{n}$ y $\boldsymbol{NL}_{v}^{n}$ son los términos no lineales para las ecuaciones de u y v respectivamente. Los operadores δ_x y δ_y realizan la derivada primera de la forma indicada en el apartado §5.6.3. Las matrices de derivada segunda son *sparse*, es decir, la mayor parte de sus elementos son cero y para ahorrar memoria no se almacena la matriz entera sino sólo aquellos elementos no nulos.

5.6.5 Condiciones de contorno

Las condiciones de contorno se pueden imponer directamente en aquellos nodos que caen sobre el contorno, sin embargo, al ser la malla staggered, el contorno no puede caer sobre nodos que contengan las velocidades u y v a la vez y por ello una de las velocidades se queda sin especificar (ver figura 5.4). Para solucionar el problema y poder imponer todas las condiciones en cada contorno, utilizaremos *celdas fantasma* tal y como se explica en la sección

§5.4. La idea consiste en poner nodos extra fuera del dominio e imponer que su punto medio cumplas las condiciones de contorno.
Por otro lado, tal y como se vio en la sección §5.5 con los ejemplos 1D, las condiciones de contorno se traducen en nuevos términos que aparecen en las ecuaciones.

5.6.6 Código

(Archivo: cavidad.m)

```
%-------------------------------------------------%
% Problema de la Cavidad                          %
%                                                 %
% Resuelve las ecuaciones de Navier-Stokes en 2D  %
%                                                 %
% Discretizacion temporal:                        %
% Euler explicito para terminos convectivos       %
% Euler implicito para terminos viscoso           %
% Paso fraccionado para incompresibilidad         %
%                                                 %
% Discretizacion espacial:                        %
% diferencias finitas                             %
% Malla staggered                                 %
%                                                 %
%                                                 %
% Adrian 2014                                     %
%-------------------------------------------------%

clear all

disp('Problema de la cavidad 2D')

% Parametros:
nu  = 0.01;    % viscosidad
CFL = 0.5;     % condicion CFL
Lx  = 1;       % tamano de la caja en x
Ly  = 1;       % tamano de la caja en y
nx  = 60;      % numero de puntos interiores en x
ny  = 50;      % numero de puntos interiores en y
nsteps = 100;  % numero de pasos temporales
```

```
Re = 1/nu;

disp(['nu:     ',num2str(nu)])
disp(['CFL:    ',num2str(CFL)])
disp(['Lx:     ',num2str(Lx)])
disp(['Ly:     ',num2str(Ly)])
disp(['nx:     ',num2str(nx)])
disp(['ny:     ',num2str(ny)])
disp(['nsteps: ',num2str(nsteps)])

% Malla:
x     = linspace(0,Lx,nx+1);
y     = linspace(0,Ly,ny+1);
[X,Y] = meshgrid(y,x);
dx    = Lx/nx;
dy    = Ly/ny;

% Condiciones iniciales:
u = zeros(nx-1,ny);
v = zeros(nx,ny-1);

% Condiciones de contorno en las paredes:
uN = x*0+1;           vN = interpolar(x)*0; % norte
uS = x*0;             vS = interpolar(x)*0; % sur
uE = interpolar(y)*0; vE = y*0;             % este
uO = interpolar(y)*0; vO = y*0;             % oeste

% Terminos procedentes de las condiciones de contorno
% para resolver el Poisson en u y v (termino viscoso)
Ubc = ([2*uS(2:end-1)' zeros(nx-1,ny-2) 2*uN(2:end-1)']/dx^2+...
    [uO; zeros(nx-3,ny); uE]/dy^2);
Vbc = ([vS' zeros(nx,ny-3) vN']/dx^2+...
    [2*vO(2:end-1); zeros(nx-2,ny-1); 2*vE(2:end-1)]/dy^2);

% Matrices para la derivada Poisson:
% Poisson para la presion
Lp      = kron(speye(ny),coef(nx,dx,1)) + ...
          kron(coef(ny,dy,1),speye(nx));  % Dxx+Dyy
% elimina la singularidad en la ecuacion
% para p, 3/2 es arbitrario
Lp(1,1) = 3/2*Lp(1,1);
```

```
% Poisson para la velocidad u
Iu = speye((nx-1)*ny);                % identidad
Lu = kron(speye(ny),coef(nx-1,dx,2)) + ...
     kron(coef(ny,dy,3),speye(nx-1)); % Dxx+Dyy

% Poisson para la velocidad v
Iv = speye(nx*(ny-1));                % identidad
Lv = kron(speye(ny-1),coef(nx,dx,3)) + ...
     kron(coef(ny-1,dy,2),speye(nx)); % Dxx+Dyy

% Inicio del bucle temporal
dt = 1e-5;
t  = 0;
for i = 1:nsteps

    % Terminos convectivos explicitos
    ue = [uO; u; uE];
    ve = [vS' v vN'];
    ue = [2*uS'-ue(:,1) ue 2*uN'-ue(:,end)];
    ve = [2*vO-ve(1,:); ve; 2*vE-ve(end,:)];
    ua = interpolar(ue')';
    va = interpolar(ve);

    uvx = diff(ua.*va)/dx;
    uvy = diff((ua.*va)')'/dy;

    ua = interpolar(ue(:,2:end-1));
    va = interpolar(ve(2:end-1,:)')';

    u2x = diff(ua.^2)/dx;
    v2y = diff((va.^2)')'/dy;

    u = u-dt*(uvy(2:end-1,:)+u2x);
    v = v-dt*(uvx(:,2:end-1)+v2y);

    % Terminos viscosos implicitos
    % ecuacion para u: (I-Dxx-Dyy)u = rhs
    ubc = dt/Re*Ubc;
    rhs = reshape(u+ubc,[],1);
    U   = (Iu-dt/Re*Lu)\rhs;
```

```
    u   = reshape(U,nx-1,ny);
    % ecuacion para v: (I-Dxx-Dyy)v = rhs
    vbc = dt/Re*Vbc;
    rhs = reshape(v+vbc,[],1);
    V   = (Iv-dt/Re*Lv)\rhs;
    v   = reshape(V,nx,ny-1);

    % Proyeccion de velocidad (Dxx+Dyy)p = ux+vy
    du  = diff([uO;u;uE])/dx;
    dv  = diff([vS' v vN'])'/dy;
    rhs = reshape(du+dv,[],1);
    P   = Lp\rhs;
    p   = reshape(P,nx,ny);
    u   = u-diff(p)/dx;
    v   = v-diff(p')'/dy;

    t   = t + dt;

    % Paso de tiempo convectivo
    ue  = [uO; u; uE];
    ve  = [vS' v vN'];
    dtx = CFL*min(dx./abs(ue(:)));
    dty = CFL*min(dy./abs(ve(:)));
    dt  = min(dtx,dty);

    % Pintar solucion
    if mod(i,1000)==0,
        disp(i)
        ue = [uS' interpolar([uO;u;uE]')'  uN'];
        ve = [vO; interpolar([vS' v vN']); vE];
        L  = 5;
        quiver(x,y,ue',ve',L,'k-')
        xlabel('x'),ylabel('y')
        axis equal, axis([0 Lx 0 Ly])
        drawnow
    end

end

disp('Fin')
```

(Archivo: interpolar.m)

```
function B = interpolar(A,k)
if nargin<2, k = 1; end
if size(A,1)==1, A = A'; end
if k<2, B = (A(2:end,:)+A(1:end-1,:))/2; else, B = avg(A,k-1); end
if size(A,2)==1, B = B'; end
```

(Archivo: coef.m)

```
function A = coef(n,h,a11)
% a11: Neumann=1, Dirichlet=2
A = -spdiags([-1 a11 0;ones(n-2,1)*[-1 2 -1];0 a11 -1],-1:1,n,n)'/h^2;
```

Agradecimientos

Quiero agradecer a Guillem Borrell y Miguel Hermanns sus valiosos comentarios que me han sido de gran ayuda en la preparación de estas notas. También estoy profundamente agradecido a todos aquellos alumnos que han seguido estas notas y se han molestado en avisarme de las erratas e inconsistencias que han encontrado.

Bibliografía

[1] R. W. Hamming Numerical Methods for Scientists and Engineers. *Dover Publications.* 1987

[2] J. L. Hennessy and D. A. Patterson Computer Architecture, Fifth Edition: A Quantitative Approach. *Morgan Kaufmann.* 2007

[3] W. Stallings Computer Organization and Architecture. 9th Edition. *Prentice Hall.* 2012

[4] P. Pacheco An Introduction to Parallel Programming. Morgan Kaufmann. 2011

[5] D. Rivas and C. Vázquez Cálculo numérico I. *Publicaciones de la Escuela Técnica Superior de Ingenieros Aeronaúticos.* 2006

[6] P. Moin Fundamentals of Engineering Numerical Analysis. Cambridge University Press. 2010

[7] J. A. Hernández Cálculo numérico en ecuaciones diferenciales ordinarias. Aula Documental de Investigación. 2000

[8] J. C. Tannehill, D. A. Anderson, R. H. Pletcher Computational fluid mechanics and heat transfer. Taylor & Francis. 1997

[9] J. H. Ferziger and M. Perić Computational methods for fluid dynamics. Springer. 2002

[10] J.D. Lambert Numerical Methods for Ordinary Differential Systems. John Wiley & Sons Ltd. 1991

[11] C. Hirsch Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics. Butterworth-Heinemann. 2007

[12] B. Seibold Course 18.086: Computational Science and Engineering I and II. Massachusetts Institute of Technology. 2007